Matthias Karlbauer

Artificial Neural Networks for Knowledge Extraction in Spatiotemporal Dynamics and Weather Forecasting

TLP-Dissertationen/Informatik

Artificial Neural Networks for Knowledge Extraction in Spatiotemporal Dynamics and Weather Forecasting

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Matthias Karlbauer
aus Regensburg

Tübingen
2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	22.03.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Martin V. Butz
2. Berichterstatter:	Prof. Dr. Georg Martius

ARTIFICIAL NEURAL NETWORKS FOR KNOWLEDGE EXTRACTION IN SPATIOTEMPORAL DYNAMICS AND WEATHER FORECASTING

Matthias Karlbauer

ARTIFICIAL NEURAL NETWORKS FOR KNOWLEDGE EXTRACTION IN SPATIOTEMPORAL DYNAMICS AND WEATHER FORECASTING

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

TÜBINGEN
LIBRARY PUBLISHING

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie, detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Die Online-Version dieser Publikation ist auf dem Repositorium der Universität Tübingen frei verfügbar (Open Access).
http://hdl.handle.net/10900/156815
http://nbn-resolving.de/urn:nbn:de:bsz:21-dspace-1568159
http://dx.doi.org/10.15496/publikation-98147

Tübingen Library Publishing 2025
Universitätsbibliothek Tübingen
Wilhelmstraße 32
72074 Tübingen
druckdienste@ub.uni-tuebingen.de
https://tlp.uni-tuebingen.de

ISBN (Druck): 978-3-98944-025-8
ISBN (PDF): 978-3-98944-024-1

ORCID Matthias Karlbauer: 0000-0002-4509-7921

Umschlaggestaltung: Susanne Schmid, Universitätsbibliothek Tübingen
Satz: Matthias Karlbauer
Druck und Bindung: Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg
Printed in Germany

Abstract

The success of meteorology and traditional numerical weather prediction (NWP) is based on more than 100 years of fundamental research in atmospheric sciences, physical differential equations, numerical simulation, chemistry, geosciences and numerous related topics. Yet, NWP has always benefited largely from advances in computing power; and the recent rise of machine learning (ML) opens avenues for revolutionary developments.

In this thesis, the potential of ML methods for improving weather forecasts are explored, contrasted, and demonstrated. Since weather is considered a spatiotemporal process, i.e., evolving over space through time, we first investigate the design choices required for ML models to simulate synthetic spatiotemporal processes—such as the two-dimensional wave equation. We then develop a method for analyzing ML models that enables the extraction of unknown process-relevant context that parameterizes an observed simulated spatiotemporal process of interest. Relating these extracted factors to physical properties leads us to physics-aware ML, where we explore how to fuse process knowledge from physics with the learning ability of artificial neural networks (ANNs). Given the insights from those investigations, we design a competitive Deep Learning Weather Prediction (DLWP) model to understand which design choices support data-driven algorithms to learn a meaningful function that predicts realistic and stable states of the atmosphere over hundreds of hours, days, and weeks into the future.

Among numerous outcomes, we identify two core factors as a basis for data-driven weather forecasting: First, the ML methods depend largely on a careful model design that cleverly incorporates inductive biases to guide the algorithm towards a physically plausible behavior. Second, data selection and preparation plays an elementary role and depends essentially on the variable to be predicted, e.g., air temperature, geopotential, or precipitation. Advances in computing hardware will allow today's two dimensional DLWP models to be extended into the third dimension. Nevertheless, we find that this hunger for compute power can be mitigated by clever model design, featuring implicit and explicit physical reasoning.

In an extensive outlook, we outline avenues for future research to advance the field of data-driven weather prediction, giving rise to more reliable forecasts that will considerably impact short- and long-term planning horizons in various domains.

Kurzfassung

Erfolge der Meteorologie und der traditionellen numerischen Wettervorhersage (NWP) beruhen auf mehr als 100 Jahren Grundlagenforschung in den Atmosphärenwissenschaften, physikalischen Differentialgleichungen, numerischer Simulation, Chemie, Geowissenschaften und zahlreichen verwandten Themen. Dabei hat NWP schon immer von Fortschritten der Rechenleistung moderner Computer profitiert; doch der jüngste Aufstieg von maschinellem Lernen (ML) eröffnet Wege für neue Entwicklungen.

In dieser Arbeit wird das Potenzial von ML-Methoden zur Verbesserung von Wettervorhersagen erforscht, gegenübergestellt und demonstriert. Da das Wetter allerdings als raumzeitlicher Prozess betrachtet wird, d.h. sich über den Raum und die Zeit hinweg entwickelt, untersuchen wir zunächst die erforderlichen Designentscheidungen mithilfe derer ML-Modelle die zweidimensionale Wellengleichung simulieren können, einen synthetischen raumzeitlichen Prozess. Anschließend entwickeln wir eine Methode zur Extraktion unbekannter prozessrelevanter Zusammenhänge aus ML-Modellen, die einen beobachteten und simulierten raumzeitlichen Prozess von Interesse parametrisieren und lokal modifizieren. Indem wir die extrahierten Faktoren mit physikalischen Eigenschaften in Beziehung setzen, gelangen wir zum sogenannten physikbewussten ML, wobei wir untersuchen, wie künstliche neuronale Netze (ANNs) Differentialgleichungen ergänzen können, indem wir das Beste aus zwei Welten kombinieren: das Prozesswissen aus der Physik und die Lernfähigkeit von ANNs. Mit den Erkenntnissen aus diesen Unter-

suchungen entwerfen wir schließlich ein konkurrenzfähiges datengetriebenes Wettervorhersagemodell (DLWP), um zu verstehen, welches Design datengetriebenen Algorithmen dabei unterstützt, eine aussagekräftige Funktion zu erlernen, die realistische und stabile Zustände der Atmosphäre über hunderte von Stunden, Tagen und Wochen vorhersagt.

Neben zahlreichen Ergebnissen identifizieren wir zwei Kernfaktoren, welche die Grundlage für erfolgreiche datengesteuerte Wettervorhersage bilden: Erstens hängen die ML-Methoden grundlegend von einem sorgfältigen Modelldesign ab, das clevere induktive Annahmen einbezieht, um den Algorithmus in Richtung eines physikalisch plausiblen Verhaltens zu lenken. Zweitens spielt die Datenauswahl und -aufbereitung eine zentrale Rolle und hängt wesentlich von der Variable ab, die Ziel der Vorhersage ist, z.B. Lufttemperatur oder Niederschlag. Ähnlich wie bei NWP gehen Verbesserungen im DLWP mit Fortschritten bei der Rechenhardware einher, um die heutigen zweidimensionalen DLWP-Modelle in die dritte Dimension zu erweitern. Unseren Beobachtungen zufolge kann dieser Hunger nach Rechenleistung jedoch durch ein geschicktes Modelldesign mit impliziten und expliziten physikalischen Annahmen kompensiert werden.

In einem ausführlichen Ausblick skizzieren wir schließlich Wege für zukünftige Forschungsrichtungen, um den Bereich der datengesteuerten Wettervorhersage voranzubringen und so zuverlässigere Wettervorhersagen zu ermöglichen, welche sowohl kurz- als auch langfristige Planungshorizonte in verschiedensten Bereichen erheblich beeinflussen und verbessern wird.

Contents

List of Figures

Acronyms

Chapter 1

Foreword

When I finished my master's thesis in the domain of autonomous driving in 2018, I decided to channel my efforts to develop ML models that are of value for society by contributing to solutions that counteract the climate crisis. My supervisor, Prof. Dr. Martin Butz, shared my passion and instantiated a project about weather forecasting with ML, which I became excited about and joined with great expectations, entering my PhD journey.

To this day, my excitement about the topic has not diminished. To be honest, however, engaging in this topic was a fairly naive enterprise, since neither I, nor my valued supervisor, had enjoyed any training in atmospheric sciences or meteorology. Instead, we considered ourselves as cognitive scientists, with a solid background in computer science and psychology with expertise in simulating cognitive processes. But how do cognitive processes relate to weather forecasting and how did we dare dive into atmospheric science?

Even though it may not seem apparent on first glance, cognitive processes share a fair bit of similarity with weather dynamics. Both the neural activities in biological brains and particle movements in the atmosphere adhere to physical principles on spatiotemporal scales. Most importantly, time-series forecasting is a common ground shared by these two disciplines.

To fuse our expertise from time-series forecasting with domain knowledge from meteorology, we reached out to numerous institutions from atmospheric sciences on the national and international level, and established dense and fruitful collaborations, which allowed me to rigorously advance my knowledge in physical process simulation related to modeling environmental processes, and to extend my skills in applying deep learning (DL) methods to exceptionally large data sets.

Despite the challenges that we were confronted with, we gained insights of high value during our journey by exploring what ML can offer for weather forecasting. These insights are condensed and presented in this thesis, as a result of almost five years of intense research in a respectful and supportive research environment.

Chapter 2

List of Publications

2.1 Publication I: Modeling Spatiotemporal Wave-Dynamics with a Location Invariant RNN

Title A Distributed Neural Network Architecture for Robust Non-Linear Spatio-Temporal Prediction (Karlbauer *et al.*, 2020a).

Journal Published in the proceedings of the 28th European Symposium on Artificial Neural Networks (ESANN), Computational Intelligence and Machine Learning, Bruges, Belgium October 02-04, 2020.

Summary A custom recurrent neural network, sharing weights when casted along spatial positions in a homogeneous grid, outperforms numerous established deep learning models when benchmarked at predicting shallow water dynamics while generalizing well on unseen conditions.

Table 2.1: Authors and their contributions in percent to the first paper, titled *A Distributed Neural Network Architecture for Robust Non-Linear Spatio-Temporal Prediction*. Legend: S = scientific ideas, D = data generation, A&I = analysis and interpretation, W = paper writing.

Author	(position)	S	D	A&I	W
Matthias Karlbauer	(1)	30	100	70	70
Sebastian Otte	(2)	25	0	15	10
Hendrik P. A. Lensch	(3)	5	0	0	5
Thomas Scholten	(4)	5	0	0	0
Volker Wulfmeyer	(5)	5	0	0	0
Martin V. Butz	(6)	30	0	15	15

2.2 Publication II: Gradient-Based Static Context Inference to Improve Data-Driven Global Weather Forecasts

Title Latent State Inference in a Spatiotemporal Generative Model (Karlbauer *et al.*, 2021).

Journal Published in the proceedings of the 30th International Conference on Artificial Neural Networks (ICANN), Artificial Neural Networks and Machine Learning, Bratislava, Slovakia, September 14-17, 2021.

Summary A distributed recurrent neural network is trained to predict global air temperature and air pressure dynamics. On a two-dimensional wave equation benchmark, the model infers the

location-varying water depth purely from observing the spatiotemporal propagation scheme of unfolding wave dynamics at the water surface. Moreover, when projecting prediction errors of air temperature forecasts on a static latent feature map, the model generates structures ressembling our planet's land-sea mask to improve its own prediction of air temperature.

Table 2.2: Authors and their contributions in percent to the second paper, titled *Latent State Inference in a Spatiotemporal Generative Model*. Legend: S = scientific ideas, D = data generation, A&I = analysis and interpretation, W = paper writing.

Author	(position)	S	D	A&I	W
Matthias Karlbauer	(1)	35	80	50	65
Tobias Menge	(2)	15	20	15	10
Sebastian Otte	(3)	15	0	10	10
Hendrik P. A. Lensch	(4)	5	0	5	0
Thomas Scholten	(5)	5	0	5	0
Volker Wulfmeyer	(6)	5	0	5	0
Martin V. Butz	(7)	20	0	10	15

2.3 Publication III: Physics-Inspired Artificial Neural Networks, Designed to Model and Extend Partial Differential Equations

Title Composing Partial Differential Equations with Physics-Aware Neural Networks (Karlbauer *et al.*, 2022).

Journal Published in the proceedings of the 39th International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, Baltimore, USA, July 17-23, 2022.

Summary One- and two-dimensional partial differential equations (PDEs) are modeled with a physics-aware neural network, designed according to the finite volume method from numerical simulation theory. Pure ML models, as well as other state-of-the-art physics-aware ML models are outperformed by orders of magnitude, while requiring just a fraction of parameters and uniquely generalizing beyond known initial and boundary conditions. In a real-world diffusion-sorption experiment, the model reveals an unknown retardation factor and predicts the propagation of a contaminant through saturated clay more accurately than a calibrated physical model.

Table 2.3: Authors and their contributions in percent to the third paper, titled *Composing Partial Differential Equations with Physics-Aware Neural Networks*. Legend: S = scientific ideas, D = data generation, A&I = analysis and interpretation, W = paper writing.

Author (position)	S	D	A&I	W
Matthias Karlbauer* (1)	40	30	40	35
Timothy Praditia* (2)	30	40	35	35
Sebastian Otte (3)	5	0	10	10
Sergey Oladyshkin (4)	5	0	0	0
Wolfgang Nowak (5)	10	30	5	10
Martin V. Butz (6)	10	0	10	10

*equal contribution

2.4 Publication IV: Predicting Global Atmospheric Dynamics with a Parsimonious Deep Learning Model on the HEALPix Mesh

Title Advancing Parsimonious Deep Learning Weather Prediction using the HEALPix Mesh (Karlbauer *et al.*, 2024).

Journal Published in the Journal of Advances in Modeling Earth Systems (JAMES) on August 20, 2024.

Summary A parsimonious U-Net is successively improved for predicting seven prognostic variables of the atmosphere. Key modifications to a previous U-Net—operating on global weather data projected to the cubed-sphere—include the inversion of channels, the incorporation of recurrent units, and the transition to the HEALPix. Despite the limited set of variables, the model falls behind state-of-the-art NWP only by a single day when generating a forecast out to one week. In contrast to sophisticated ML approaches, the model can be unrolled autoregressively for hundreds of time steps, generating realistic states of the atmosphere and retaining seasonal trends.

Table 2.4: Authors and their contributions in percent to the fourth paper, titled *Advancing Parsimonious Deep Learning Weather Prediction using the HEALPix Mesh*. Legend: S = scientific ideas, D = data generation, A&I = analysis and interpretation, W = paper writing.

Author	(position)	S	D	A&I	W
Matthias Karlbauer	(1)	35	80	40	50
Nathaniel Cresswell-Clay	(2)	10	10	15	0
Dale R. Durran	(3)	40	0	35	40
Raúl A. Moreno	(4)	0	0	5	0
Thorsten Kurth	(5)	5	0	0	0
Boris Bonev	(6)	0	5	0	0
Noah Brenowitz	(7)	0	5	0	0
Martin V. Butz	(8)	10	0	5	10

Chapter 3

Introduction

Weather is the product of a soft film, no more than a few kilometers thick, gently surrounding our planet. Climate science calls this film troposphere, the lowest of five layers composing the atmosphere. At the equator, where the centrifugal forces of the Earth's rotation are greatest, the troposphere reaches an altitude of up to 15 km, while at the poles it is only half as thick, measuring approximately 7 km.

Even though the atmosphere is composed of lightweight gas molecules, it has the capacity to absorb enormous amounts of energy, which can discharge in severe lightnings, heavy storms, or pouring rain that can cause distress to any creature on the surface. Crucially, the energy that can be absorbed by the atmosphere increases with temperature (Durran and Frierson, 2013), since warm air has a higher moisture content that condensates to clouds when cooling down upon entering higher altitudes.[1] Hence, with increasing global temperatures, the energy level in the atmosphere will rise, which will, in consequence, manifest in both more frequent and more severe extreme events in future. Heavy record breaking rainfall events, for example, are now occurring more frequently all

1 With increasing altitude, the temperature in the troposphere falls by 0.5-0.75 °C/100 m, reaching about −50 °C at the tropopause, which denotes the top layer of the troposphere.

over the world (Malik *et al.*, 2020; Post and Knapp, 2020; Zhang, 2020), which might also be a result of the increasing temperature that weakens the north-south temperature gradient, resulting in more atmospheric blocking events, where weather systems are trapped in one location over longer periods (Stendel *et al.*, 2021).

Accordingly, in times of antropogenic climate change, where our planet's condition and biodiversity are increasingly challenged (Steffen *et al.*, 2015), the improvement of weather and climate forecasts plays an elementary role to determine the intensity, time, and location where an extreme event is likely going to unfold, and to derive strategies that prevent tipping points (Lenton *et al.*, 2019) from being triggered, since surpassing tipping points might lead to a system collapse.

3.1 Spatiotemporal Processes

A wide variety of processes in our world have a spatial and a temporal component, such as electric signals propagating through the neurons, dendrites, and axons in our brains and bodies, social networks that describe relations between individuals in social media, the flow of vehicles in densely populated cities described as traffic, and any kind of chemical or physical phenomenon unfolding on micro and macro scales, describing the interactions of molecules or the movement of substance within a medium.

Traditionally, spatiotemporal processes are modeled in three steps: First, the process is formulated as a PDE that describes the temporal evolution of a quantity of interest, such as temperature, under influence of certain factors, e.g., incoming solar radiation. Second, the PDE is solved for the quantity of interest, often requiring the application of discretization schemes over space and time to convert

the PDE into an ordinary differential equation (ODE).[2] Third, the resulting ODE is modeled via numerical simulation, projecting the distribution of the quantity of interest into the future. Each of these three steps—the formulation of an appropriate PDE, finding a solution, and simulating it—requires advanced process understanding.

In contrast, recent trends in computer science and ML simulate spatiotemporal processes in a data-driven manner, modeling the system's state progression without any detailed knowledge of the underlying causality.

3.2 Machine Learning

While numerical simulation techniques approach a problem from the side of process understanding by formulating an equation to simulate the dynamics of a system, ML solves the challenge from the perspective of the data. The goal of ML, therefore, is to design a model that learns to simulate the dynamics of a system merely from the observation of data.

Although ML embraces a vast number of unsupervised, semi-supervised, and supervised techniques that can be applied to solve an even wider variety of problem classes, ranging from clustering to classification to regression and temporal forecasting, we focus on supervised learning methods here to perform time-series forecasting.

Supervised learning describes the situation, where a model is exposed to a task for which the solution is known. For example and

2 While PDEs depend on multiple unknown variables, an ODE only depends on one unknown variable, which allows the determination of a unique solution.

most prominently, the classification of objects on images constitutes a supervised learning task, where the model is provided with an image as input and trained to produce a correct label of the object that is depicted on the image.

In time-series forecasting, the ML model typically receives a series of $1, \ldots, t$ inputs (with $t \in \mathbb{N}$ denoting the time index of the time series' consecutive data points) and is trained to continue the progression of the time series from $t+1, \ldots, \tau$, where $\tau \geq t+1$ is the index of the last time step of interest. Thus, in contrast to image classification, which requires human experts to annotate each image to generate an according label, the labels are provided more naturally in supervised time-series forecasting, which constitutes the continuation of a signal over time.

In the next section, some fundamental techniques for supervised time-series forecasting are summarized, followed by state-of-the-art methods that found their way into this discipline more recently.

3.2.1 Foundations of Time-Series Forecasting with ML

Feedforward Networks

A popular class of supervised ML algorithms are ANNs, which are based on the early work of McCulloch and Pitts (1943), who developed the perceptron—a model of the biological neuron that accumulates inputs and generates an output once the summed inputs surpass a particular threshold.

Rosenblatt (1958) transformed the perceptron into a continuous version, equipping it with an activation function that enables the approximation of nonlinear dynamics. Connecting individual neurons in multiple layers with weighted connections resulted in the

so called multilayer perceptron (MLP), which was later trained via the backpropagation algorithm by Linnainmaa (1970). The backpropagation algorithm computes the partial derivatives that each weighted connection between any two neurons in the model contributes to the prediction error and applies the negative weight-specific derivative to each weight, respectively. Repeating this procedure with hundreds of input-output pairs, effectively implements a gradient-descent process, which minimizes the error function progressively, essentially modifying the weights incrementally such that, for a given input, the model's output most accurately reproduces the associated label.

Recurrent Neural Networks

In feedforward networks, the information always flows from the input to the output. Accordingly, when processing time-series data sequentially—by feeding the model iteratively with consecutive data points—the model has no capacity to memorize previous inputs or states when processing the current input.

This limitation is addressed in recurrent neural networks (RNNs), which introduce loops into the model pipeline such that information from previous time steps can be taken into account when processing the current model input (Amari, 1972). Modern recurrent models are typically trained with the backpropagation through time (BPTT) algorithm, which essentially unrolls an RNN over time to apply the backpropagation algorithm to it.

When processing long time-series, however, RNNs suffer from the vanishing gradient problem (Hochreiter, 1991), where the multiplicative nature of the chainrule—employed for determining the gradient signals in the unrolled RNN to update the model's weights—effectively decreases the gradient signal when approaching earlier layers of the unrolled RNN.

To overcome this issue, Hochreiter and Schmidhuber (1997) introduced gating mechanisms to trap the error signal inside of the cell state of their so called long short-term memory (LSTM), which allowed the learning of long-term dependencies. Later, a marginally simplified version of the LSTM, named gated recurrent unit (GRU), was introduced by Chung *et al.* (2014)

3.2.2 Recent Advances in Deep Learning

Convolutional Neural Networks

An efficient way of processing homogeneously distributed data, such as images, is implemented in the convolutional neural network (CNN), which can be interpreted as a small MLP with a bounded receptive field, e.g., a 3×3 stencil, that shares weights when being cast along the spatial dimensions of the input. In CNNs, the size of the receptive field of a single convolution layer is called kernel size k, with $k = 3$ in most applications, as in the stencil example above.

When Krizhevsky *et al.* (2012) designed their CNN, which outperformed the traditional image classification models with handcrafted feature detectors by a large margin, ML recovered from its second artificial intelligence (AI) winter, initiating the third wave of AI research that benefits substantially from powerful compute hardware such as graphics processor units (GPUs) and enormous amounts of data to train according models. Incidentally, the present era of DL is considered likely to turn into a perpetual summer (Kautz, 2022).

Modeling Spatiotemporal Processes

Temporal Convolution Network CNNs are predominantly applied to two-dimensional image data. Thus, Lea *et al.* (2016)

consider a one-dimensional time-series as two-dimensional vector and redesigned the convolution operation to ignore the component of the kernel that looks into the future of the temporal data dimension. Moreover, when stacking multiple of their custom convolution layers, they apply dilations that increase with a factor of two per layer to widen the receptive field of their temporal convolutional network (TCN) more efficiently.

TCN has been demonstrated to outperform traditional recurrent models that evolve a latent state over time (Kalchbrenner *et al.*, 2016). In our experiments of research units (RUs) I through III, we adopt the original formulation of TCN to allow the processing of two-dimensional image sequences and contrast our findings with recurrent DL models.

ConvGRU and ConvLSTM Naively employing an RNN to spatiotemporal data would either result in a huge model when choosing the fully connected (FC) strategy, or prevent the model from processing spatial information when casting the RNN along spatial dimensions of the data with shared weights.

Ballas *et al.* (2015) therefore developed convGRU, which combines CNNs with the GRU cell by replacing all dot products at the cell input and gates by convolution operations to effectively provide the cell with a receptive field of arbitrary size in k. The same principle was applied to the vanilla LSTM cell by Shi *et al.* (2015), resulting in convLSTM, which proved superior to traditional methods when forecasting the progression of precipitation data from radar scans. Accordingly, we compare convLSTM with our model in the experiments conducted in RUs I through III.

DISTANA In developing our own model for spatiotemporal processing, we formulated a graph neural network (GNN) that con-

sists of two types of kernels, which are both small combinations of feedforward and LSTM components. Prediction kernels (PKs) are designed to simulate the dynamics of each node in the considered graph, while transition kernels (TKs) take the role of routing information between adjacent PKs. The advantage of our DISTANA lies in its applicability to spatially regular or distributed data.

When simulating series of regular image data, the TKs are not contributing any value and can be discarded, such that PKs exchange lateral information directly. In this formulation, DISTANA shares high similarity with convLSTM. A key difference, however, lies in the realization of lateral information exchange between adjacent pixels: While convLSTM reads the activity of neighboring cells directly from the input (by applying convolution operations at the inputs of all gates, including the cell input), DISTANA implements an explicit latent feature map that is solely designed for lateral information exchange, making it more parameter efficient and allowing to decouple lateral information flow from the propagation of target dynamics within the network.

Physics-Aware Machine Learning

When the laws of physics, governing the spatiotemporal evolution of a system, are known and formulated in a specific PDE, we may employ ANNs to parameterize these physical laws. An example from video prediction literature is the generation of motion fields with a CNN that can be used to parameterize a numerically implemented advection process (Reda *et al.*, 2018; De Bézenac *et al.*, 2019; Luc *et al.*, 2020)

In an exhaustive review about physics-aware machine learning, Karniadakis *et al.* (2021) summarize different approaches to combine physical principles with the learning ability of ANNs. Early attempts introduced physics-related terms to the loss function in

order to minimize physical properties such as divergence or conservation of mass, and to enforce smoothness in first- and second-order spatial and temporal derivatives (De Bézenac *et al.*, 2019; Raissi *et al.*, 2019; Sirignano and Spiliopoulos, 2018). Moreover, the prediction of residuals (rates of change) over absolute values is a simple yet powerful design choice, related to solving PDEs in computational physics by means of ODE solvers [(He *et al.*, 2016; Chen *et al.*, 2018).

Another group of models applies neural operators (Li *et al.*, 2020; Pathak *et al.*, 2022) to learn a class of functions to simulate the progression of a spatiotemporal process instead of learning a single function like done by Raissi *et al.* (2019). Other approaches, such as our FInite volume Neural Network (FINN), embed physical laws more directly through a dedicated network design, enforcing different modules to account for particular phenomena (Sirignano and Spiliopoulos, 2018; Yin *et al.*, 2021; Karlbauer *et al.*, 2022; Lienen and Günnemann, 2022).

Gradient-Based Inference

A key advantage of DL models lies in their differentiability, which allows the propagation of error signals—observed at the model output—towards any variable used within the model. This gradient-based inference found application in determining weak points of image classifiers by means of *adversarial attacks*, where the input to a DL classification model is manipulated marginally to change the output of the classifier (Akhtar and Mian, 2018).

Other efforts employed gradients to realize model predictive control by defining a target position of interest as the desired model output and projecting the network's prediction errors at reaching this target backwards onto the inputs of the model, which translated into motor signals of a robot. When doing so, Otte *et al.*

(2017) and Butz *et al.* (2019) effectively steered an ensemble of different robot types, which was determined via gradients as well. In Otte *et al.* (2020), we called this gradient-based inference principle active tuning (AT), and used it to eliminate noise from model inputs, proving AT as a valuable alternative to teacher forcing in situations where the model needs to be tuned into unfamiliar dynamics, such as high noise conditions that were not encountered during training.[3]

Furthermore, in RU II, we apply AT to infer static context maps that parameterize the location-specific evolution of spatiotemporal processes, such as inferring the water depth in the SWE, or revealing the land-sea mask (LSM) when predicting global air temperature dynamics with DISTANA (Karlbauer *et al.*, 2020b).

3.3 Earth System Modeling and Weather Forecasting

An Earth system model (ESM) simulates global processes in the atmosphere and oceans (including chemical and biochemical processes therein) by taking into account the effects of vegetation and the global carbon cycle. Thus, an ESM extends over global climate models (GCMs), which simulate the atmosphere alone. GCMs were originally formulated purely as a NWP method, as described below in Section 3.3.1, a trend that is increasingly questioned by recent successes of DL-based GCMs, outlined in Section 3.3.2.

3 We later replicated the suitability of AT for cleaning noisy inputs in combination with spiking neural networks (Ciurletti *et al.*, 2021).

3.3.1 Traditional Methods

The history of traditional NWP dates back to 1922, when Lewis Fry Richardson published his book *Weather Prediction by Numerical Processes* (Richardson, 1922). Richardson employed numerous computers—human beings instead of machines at the time—to solve differential equations describing the state of the atmosphere in order to predict the weather. In his experiment, the state of the atmosphere on the 20$^{\text{th}}$ of May 1910 at 7 am was selected as initial condition (IC) and a retrospective forecast was computed for 1 pm, that is for a lead time of six hours.

Although Peter Lynch later criticized Richardson's forecast of the air pressure as imprecise (the model predicted a rise of pressure by 145 hPa while the true pressure remained about constant), Lynch also embraced Richardson's pioneering work. Concretely, Lynch detected an essential but mathematically non-fundamental source of error in Richardon's calculations. When applying smoothing techniques to avoid physically implausible pressure fluctuations, Richardson's method eventually resulted in an accurate forecast (Lynch, 2006).

In the meantime, NWP did undergo a "quiet revolution" (Bauer *et al.*, 2015) and, as of today, produces skillful forecasts of the atmosphere with lead times of ten or more days. Essentially, the success of NWP is built on three pillars: First, the IC for the simulation is determined carefully during an assimilation phase by integrating various sources of information from remote sensing (satellite) to ground-based measurements. An accurate estimate of the IC is substantial, since even minor deviations from the true initial state of the weather system can inflate to large prediction errors throughout the forecast, cf. Lorenz' butterfly effect (Lorenz, 1963a,b). Second, physical processes—such as the mixture of air or water with different temperatures or concentrations (diffusion), transport of

particles through wind (advection/convection), or ground heating via radiation—are simulated by means of Navier-Stokes and mass continuity equations under consideration of the first law of thermodynamics and the ideal gas law (Bauer *et al.*, 2015). The resulting system of equations must be solved at up to 0.5 billion locations (depending on the model resolution), distributed horizontally and vertically across the globe. Third, the simulation is repeated multiple times with slightly different ICs to both account for errors in the data assimilation and forecasting phase, and to get an uncertainty estimate of the weather forecast, quantifying the likelihood of future events. Thus, the simulation of multiple forecasts leads to ensembling methods that convert the deterministic forecast of a single model into a probabilistic prognosis on basis of, e.g., 51 individual ensemble forecast members (Buizza and Leutbecher, 2015; Palmer, 2019).

However, the ever finer resolution in time and space—which is necessary to resolve small scale processes such as convection and turbulence and to further improve NWP forecasts—as well as the repeated simulation of this complex process to quantify uncertainties, imposes immense computational burdens and demands the world's largest supercomputers (Váňa *et al.*, 2017).[4]

3.3.2 Deep Learning Weather Prediction

Taking up the argument of high computational burdens, a common critique against the training of large DL models is the expensive amount of energy they depend on (Strubell *et al.*, 2019). However, according to Pathak *et al.* (2022), they required around as much energy for training their DLWP model, as a forecast over ten days with the state-of-the-art NWP ensemble system would cost. Thus

4 `https://www.metoffice.gov.uk/about-us/what/technology/supercomputer`

in the long-term run, applying DLWP will, in fact, turn out as more energy efficient compared to NWP. In particular, since inference with a trained ML model can be realized with just a fraction of the energy required for training. Kaack *et al.* (2022) further emphasizes this important difference between model development and inference: While model training is expensive, it amortizes at inference time when computations are much cheaper. Nevertheless, the rebound-effect likely causes a higher energy consumption in the end, as these energy efficient methods might be operated in larger ensembles.

The Potential of DL

Several researchers investigated the fundamental potential of DL for weather forecasting. Dueben and Bauer (2018), for example, were asking "Can models that are based on deep learning and trained on atmospheric data compete with weather and climate models that are based on physical principles and the basic equations of motion?" Similarly, Weyn *et al.* (2019) and Schultz *et al.* (2021) titled their works "Can machines learn to predict weather?" and "Can deep learning beat numerical weather prediction?", respectively.

In agreement with Scher and Messori (2018); Reichstein *et al.* (2019), all authors come to the conclusion that DLWP has indeed great potential for weather forecasting, albeit still falling far behind the state-of-the-art NWPs from ECMWF. While White *et al.* (2017) see the greatest limitations of NWP in the subseasonal-to-seasonal (S2S) range, that is, at lead times between two to six weeks, Dueben and Bauer (2018) suggest the application of DLWP to medium-ranged forecasts, going out to lead times of up to two weeks. In particular, Dueben and Bauer (2018) are concerned with the physical plausibility of forecasts that are generated by DL mod-

els that do not implement any physical constraints. In fact, DLWP models are not guaranteed to behave physically plausible, particularly in out of distribution domains, when the model is applied to conditions that were not part of the training.

Since 2022, the potential of pure DL models for weather prediction has been increasingly demonstrated by commercial tech-enterprises. As such, NVIDIA's FourCastNet (Pathak *et al.*, 2022), Google's GraphCast (Lam *et al.*, 2022), Huawai's Pangu-Weather (Bi *et al.*, 2023), and Microsoft's ClimaX (Nguyen *et al.*, 2023) are just about to surpass the skill of NWP's state-of-the-art by employing ever increasing numbers of variables (Ben-Bouallegue *et al.*, 2023). Despite their impressive forecast accuracy, though, these models remain opaque and have not yet contributed to an improved process understanding. Thus, it is of high scientific interest to investigate how competitive DLWP models generate accurate forecasts of the atmosphere and whether the learned functions match those from traditional NWP. Regardless of whether they do or not, it would be a revealing outcome in any case. Essentially, the potential of DL for weather prediction not only lies in an improved forecast quality, but also in knowledge gain of atmospheric processes that are not yet understood or known.

Hybrid Formats of Numerical and Deep Learning Weather Prediction

In global atmospheric science, physics-informed neural networks have hardly been applied so far; see Kashinath *et al.* (2021) for a review on first steps and early successes. Pioneering works mostly relate to subgrid parameterizations, where ML is applied to learn a mapping of high-resolution-dependent processes, such as convection, to lower resolutions (Rasp and Lerch, 2018; Beucler *et al.*, 2019; Gentine *et al.*, 2020; Beucler *et al.*, 2021; Yuval *et al.*, 2021).

This allows the operation of a physical model on coarse resolution without losing essential convective information, as it is contained in the parameterization through the neural network.

Moreover, most recent works by Bauer *et al.* (2023); Shen *et al.* (2023); Tesch *et al.* (2023) further suggest the combination of meaningful physical constraints with the learning ability of ANNs, also promoting a compositional structure, such as in our FINN, which is detailed in RU III, Section 4.3.

3.4 Objective and Expected Outcome of Doctoral Research

In this thesis, we strive to answer four central research questions, outlined in the following. Four associated RUs are designed to answer each of these questions, as detailed in Section 4.1, Section 4.2, Section 4.3, and Section 4.4, respectively.

(1) **Question:** How can spatiotemporal processes be modeled with ML models in a purely data driven setup and what are the necessary architectural inductive biases?

Expected Outcome: The successful simulation of an exemplary spatiotemporal process with an ML approach and a breakdown of the relevant architectural features for the successful simulation.

(2) **Question:** How can unknown parameterizations of spatiotemporal processes be extracted purely from the observation of the evolution of the process's temporal dynamics?

Expected Outcome: Extract process-relevant features

that affect the temporal evolution of a spatiotemporal process and relate the extracted features to physical principles.

(3) **Question:** What is the role and relevance of physical constraints in ML models, i.e., in physics-aware ANNs, and how do they contribute to the simulation of spatiotemporal processes?

Expected Outcome: Understand how ANNs can augment physical process equations, i.e., PDEs, and determine what advantages and limitations physics-aware ANNs have when compared to pure ML models.

(4) **Question:** What are the key ingredients for a DLWP model that enables the generation of stable and physically plausible predictions over long lead times?

Expected Outcome: Deployment of a competitive DLWP model with recurrent connections that supports hundreds of autoregressive steps without collapsing activities to help identify key design choices.

Chapter 4

Results and Discussion

This chapter is divided into four sections featuring four RUs, with each RU addressing the research objective that has been pursued in one of the four publications included in this thesis. We focus on results and discuss them in the broader scope of this work. Model details and additional information can be found in the respective publications, which are supplemented in Appendix A. To review the intent, objective, and expected outcome of each individual RU, the reader is referred to Section 3.4.

4.1 Research Unit I: DISTANA

4.1.1 Results

Dataset I: Dampened Sinusoidal

Data Description The first dataset implements a dampened sinusoidal on a 16×16 pixel grid with varying amplitude, which depends on the distance to the center of a circularly outwards expanding wave (see Figure 4.1 for an illustration). Since quantity is not effectively propagating laterally, this dataset is considered comparably simple. Technically, the models have to simulate a sine with decaying amplitude per location. The challenge, how-

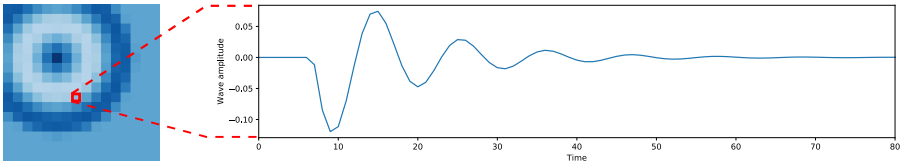


Figure 4.1: Depiction of a sample from Dataset I (simple). Left: exemplary circular wave in timestep 12. Right: activity pattern over time at one particular position in the two-dimensional wave field. Figure and caption modified from (Karlbauer *et al.*, 2020a).

ever, lies in simulating the correct onset in each location and the activity in form of the right amplitude and its time derivative (rising or falling) so that the impression of a spatially propagating wave emerges. This requires the models to properly initiate the sine oscillation in each location with the correct dynamic intensity and direction, which substantially depends on the spatial activity progression in the adjacent cells.

Model Selection Six model categories are contrasted and evaluated. Categories include (1) baselines, (2) FC ANNs operating on flattened tensors, (3) CNNs without temporally evolving latent states, (4) CNN-LSTM-combinations, (5) RNN variants that process image pixels sequentially, and (6) GNNs such as our DISTANA. Quantitative results and root mean squared error (RMSE) scores are reported in Table 4.1 and put in context of each model category below.

(1) **Baselines:** Two baselines are implemented and tested, reported as *Baseline zero* and *Baseline* $t-1$. While the former always produces zeros, the latter (also called *persistence* in the time-series forecasting literature) predicts the last observed state as a constant value for all remaining time steps.

On the regular test set with constant wave amplitude and

Table 4.1: Results on Dataset I (simple), showing training and test set errors as RMSE along with inference time in seconds. *1-train-ex.* and *Var. wave* columns show test errors when trained on a single example and on waves with variable amplitude, respectively. Best results in bold. Table modified from Karlbauer *et al.* (2020a).

Model (#pars)	Train error	Test error	1-train-ex.	Var. wave	Inf. time
Baseline $t-1$	-	3.59×10^{-5}	3.59×10^{-5}	5.90×10^{-5}	-
Baseline zero	-	8.88×10^{-5}	8.88×10^{-5}	5.84×10^{-4}	-
FC-Linear (65k)	2.36×10^{-4}	2.56×10^{-4}	2.41×10^{-4}	1.91×10^{-3}	**0.0051**
FC-LSTM (524k)	5.34×10^{-5}	1.38×10^{-2}	2.14×10^{-3}	6.57×10^{-3}	0.0113
CNN (20)	3.41×10^{-4}	2.22×10^{-4}	1.37×10^{-3}	2.66×10^{-2}	0.0115
TCN (2.3k)	1.17×10^{-5}	8.56×10^{-3}	1.04×10^{-1}	3.09×10^{-2}	0.0531
CLSTMC (768k)	6.28×10^{-5}	4.67×10^{-1}	6.13×10^{-4}	2.71×10^{-1}	0.0230
ConvLSTM1 (144)	1.83×10^{-5}	4.26×10^{-5}	4.55×10^{-5}	5.85×10^{-4}	0.0247
ConvLSTM8 (2.9k)	6.34×10^{-6}	$\mathbf{1.28 \times 10^{-6}}$	7.88×10^{-4}	1.29×10^{-2}	0.0298
GridLSTM (624)	7.95×10^{-5}	3.62×10^{-1}	2.86×10^{-1}	1.35×10^{-1}	5.8786
BiGridLSTM (1.8k)	$\mathbf{6.28 \times 10^{-6}}$	5.65×10^{-1}	8.67×10^{-1}	4.55×10^{-2}	11.9900
DISTANA4 (108)	4.18×10^{-5}	2.08×10^{-5}	1.41×10^{-5}	2.17×10^{-4}	0.0264
DISTANA26 (2.9k)	2.58×10^{-5}	1.48×10^{-5}	$\mathbf{2.04 \times 10^{-5}}$	$\mathbf{9.99 \times 10^{-5}}$	0.0326

speed, *Baseline zero* and *Baseline* $t-1$ yield mean squared error (MSE) scores of 8.88×10^{-5} and 3.59×10^{-5}, respectively. On the test set with varying wave amplitudes and speed, the scores rate 5.84×10^{-4} and 5.90×10^{-5} .

These baselines must be outperformed in order to call the predictions of a respective model skillful.

(2) **FC-ANNs:** Flattening the two-dimensional 16×16 pixel grid into a one-dimensional tensor with 256 entries allows the application of FC layers. Neurons in such layers receive input from all 256 pixels to predict the dynamics at each pixel in the next time step. Two models are evaluated: A *FC-Linear* layer with 256 neurons, and a *FC-LSTM* layer consisting of 256 LSTM cells.

With MSE scores of 2.56×10^{-4} and 1.38×10^{-2} on the regular test set, neither the *FC-Linear* nor the *FC-LSTM* model reaches the baseline accuracy. The same is observed for the

specific cases, when the models are trained on a single sample only (2.41×10^{-4} and 2.14×10^{-3}) or when trained and tested on variable wave dynamics (1.91×10^{-3} and 6.57×10^{-3}).

(3) **Pure Convolution Approaches:** CNNs are parameter-efficient models by sharing weights when being cast along spatial dimensions. Next to a single-layer *CNN* with kernel size 3×3 (other design choices with more layers and different kernel sizes were tested as well), a *TCN* is considered, consisting of three layers with $[1, 8, 1]$ feature maps and kernel size $3 \times 3 \times 3$.

Similar to the FC approaches, convolution models fall behind the baselines with MSE scores of 2.22×10^{-4} and 8.56×10^{-3} on the normal test set (for *CNN* and *TCN*, respectively), 1.37×10^{-3} and 1.04×10^{-1} in the single-sample condition, and 2.66×10^{-2} and 3.09×10^{-2} on the variable waves.

(4) **Convolution-LSTM Combinations:** A single-layer convolution encoder and decoder with an LSTM between is implemented (*CLSTMC*) and contrasted with two single-layer convLSTM models with one (*ConvLSTM1*) and eight (*ConvLSTM8*) convLSTM cells, respectively.

While the naive *CLSTMC* falls short of the baselines (4.67×10^{-1}, 6.13×10^{-4}, and 2.71×10^{-1} in the three conditions), the convLSTM models generate partially skillful predictions. On the regular test set, the *ConvLSTM1* score (4.26×10^{-5}) lies between the two baselines, while *ConvLSTM8* reaches a remarkably small error with an MSE of 1.28×10^{-6}. In the single sample condition, *ConvLSTM1* (4.55×10^{-5}) again lies between the two baselines, however, *ConvLSTM8* (1.29×10^{-2}) falls far behind. On the varying waves, both models perform slightly worse than *Baseline zero* with 5.85×10^{-4} and 7.88×10^{-4}.

(5) **Sequential recurrent neural networks (RNNs):** Two sequential RNN models are evaluated, namely a *GridLSTM* (Kalchbrenner *et al.*, 2015), which runs forward in time and space, and its successor, *BiGridLSTM* (Fei and Tan, 2018), running forward in time and bidirectional in space.

Although both methods achieve remarkable training errors, they perform poorly on the three test cases. Concretely, *GridLSTM* produces MSE scores of 3.62×10^{-1}, 2.86×10^{-1}, and 1.35×10^{-1}, while *BiGridLSTM* scores 5.65×10^{-1}, 8.67×10^{-1}, and 4.55×10^{-2}.

(6) **GNNs:** Our DISTANA is evaluated in two different size configurations. *DISTANA4* is equipped with four LSTM cells in the PK and *DISTANA26* has 26 of these LSTM cells. Due to the homogeneous grid on which the data is represented, we treat transition kernels (TKs) as identity function.[1]

On the normal test set, *DISTANA4* and *DISTANA26* score 2.08×10^{-5} and 1.48×10^{-5}, respectively, both outperforming the baselines. We make the same observation in the single sample condition, with unique MSE scores of 1.41×10^{-5} and 2.04×10^{-5}. On the varying wave dataset, both models outperform *Baseline zero* (2.17×10^{-4} and 9.99×10^{-5}) without reaching the level of *Baseline t-1*.

1 Note that graph neural networks (GNNs) can operate on spatially regular and irregularly distributed data. The regularity of image data considered here eliminates the need to dynamically treat neighbors with widely varying distances from different directions, e.g., by employing TKs. We have experimented with irregularly distributed data points by removing up to 75% of PKs from the mesh and enabling lateral communication via TKs. When providing each TK with the distance and angle between two PKs, DISTANA proved capable of simulating the propagating of the circular wave dynamics in a sparsely populated grid among irregularly distributed PKs.

Dataset II: Two-Dimensional Wave Equation

Data Description The previous dataset, presented and evaluated in Section 4.1.1, neither propagates quantity laterally explicitly, nor does it exhibit complex and interactive wave patterns. We thus generate a second dataset by recruiting the two-dimensional wave equation, a PDE that we solve by discretizing over space and time with the finite difference method, as detailed in our extended preprint, which is available on `arXiv`.[2]

We apply a Dirichlet boundary condition (BC) by setting the values of hypothetical neighbors outside of the simulation domain to zero. This causes waves approaching a border of the simulation domain to be reflected back into the field. Resulting dynamics mirror the water ripples in a swimming pool, as illustrated in Figure 4.2.

Model Selection In accordance with the findings of our previous evaluation series, we confine ourselves to the three most promising model architectures and benchmark *TCN*, *ConvLSTM8*, and three refinements of DISTANA, detailed below, against the established baselines. Results are reported in Table 4.2, visualized in

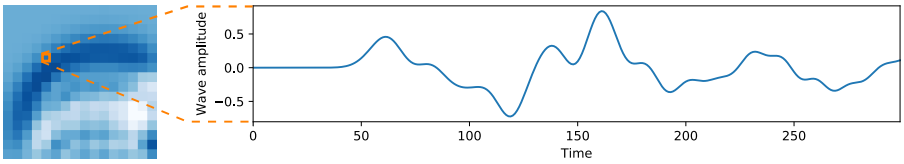


Figure 4.2: Left: 16 × 16 data grid example, showing a circular wave sample from Dataset II (complex) around time step 55 propagating from bottom right to top left and being reflected at the borders. Right: the wave amplitude dynamics over time for one pixel in the 2D wave field. Figure and caption modified from Karlbauer *et al.* (2020b).

2 `https://arxiv.org/abs/1912.11141`

Table 4.2: Results on Dataset II (complex), showing training and test set errors as RMSE along with inference time. Best results in bold. Table modified from Karlbauer *et al.* (2020a).

Model (#pars)	Train error	Test error	Inf. time
Baseline $t-1$	-	5.83×10^{-3}	-
Baseline zero	-	1.07×10^{-2}	-
TCN (2.3k)	1.14×10^{-5}	2.11×10^{-1}	0.0707 s
ConvLSTM8 (2.9k)	3.52×10^{-6}	8.09×10^{-2}	0.0289 s
DISTANAv1 (146)	7.89×10^{-6}	8.77×10^{-3}	**0.0280**
DISTANAv2 (172)	$\mathbf{1.36 \times 10^{-6}}$	7.68×10^{-4}	0.0294
DISTANAv3 (200)	1.64×10^{-6}	$\mathbf{4.99 \times 10^{-}4}$	0.0301

Figure 4.2, and detailed in the following.

(1) **Baselines:** MSE scores of *Baseline zero* and *Baseline* $t-1$ on the test set of the second and more challenging dataset amount to 1.07×10^{-2} and 5.83×10^{-3}.

(2) **TCN:** As in the first experiment, *TCN* performs worse than both baselines, generating an MSE of 2.11×10^{-1}. This poor score is well supported by the examplary dark blue line in Figure 4.2, where *TCN*'s prediction leaves the reasonable value range around time step 62.

(3) **convLSTM:** In contrast to the prior series of experiments, *ConvLSTM8* does not lie in between the two baselines, but falls behind when producing a test error of 8.09×10^{-2}. As emphasized in the orange line in Figure 4.2, *ConvLSTM8*'s predictions oscillate heavily, taking amplitudes beyond the reasonable value range.

(4) **DISTANA:** Three variants of DISTANA are evaluated to

showcase effects of model design when simulating realistic wave dynamics.

DISTANAv1 corresponds to *DISTANA4* as configured in the previous experiments, apart from the preprocessing tanh-layer in the PK, which is increased from two to four neurons. Lateral information from all eight adjacent neighbors is summed in one single lateral input neuron. Producing an MSE score of 8.77×10^{-3}, this model lies well between the baselines.

DISTANAv2 extends *DISTANAv1* by introducing eight dedicated lateral input neurons, selectively receiving inputs from each respective neighbor. With one input neuron per neighbor, this architecture is not direction-invariant anymore, but treats directional information explicit by allowing direction-sensitive weights to evolve. We observe this augmentation to improve the performance meaningfully, reducing the error to 7.68×10^{-4} and thus outperforming both baselines.

DISTANAv3 further enhances *DISTANAv2* by also increasing the lateral output neurons from one to eight. These neurons are designed to selectively route direction-specific information to the respective neighbors. With this modification, the prediction skill is further advanced, allowing the model to reach an MSE of 4.99×10^{-4}.

4.1.2 Discussion

We contrasted various ANN architectures against sophisticated baselines on two datasets resembling the spatiotemporal evolution of a two-dimensional circular wave that propagates outward from a point source. Our own model, DISTANA, has been evaluated under numerous configurations and achieved superior scores on

the MSE metric, supported by qualitative results. In particular, DISTANA learned to simulate the dynamics accurately with very few parameters and uniquely demonstrated robust generalization on both unseen and variable data. Key observations and insights of this RU are discussed in the following.

The Central Role of Model Design Even though Hornik *et al.* (1989) theoretically proved MLPs as universal function estimators, we can confirm that ANNs benefit substantially from a careful architecture design. A task-sensible design—referred to as implementing reasonable relational inductive biases Battaglia *et al.* (2018)—facilitates the evolution of a function purely from data by constraining the function space meaningfully, thereby reducing the search space that stands for exploration.

We witnessed the relevance of model design several times during the conducted research:

- Only a very small portion of models—ranging from FC ANNs, pure CNNs, convolution-LSTM combinations, sequential RNNs, to GNNs—were able to simulate the evolution of two-dimensional wave dynamics at all over multiple autoregressive time steps into the future.
- Even though TCN by design has the spatial and temporal capacity required to evolve two-dimensional dynamics into the future, which is emphasized by reaching smaller training errors compared to DISTANA, it performed poorly when operating in closed loop. Likely, the lack of a latent hidden state—evolving over time in recurrent models and formulating a stable attractor of the currently ongoing dynamics—prevents TCN from a successful autoregressive progression. Arguably, TCN might be better suited for efficient temporal information aggregation, making it a promising candidate for

data compression and encoding.

- The presence of a latent recurrent state alone, however, seemingly is not sufficient for a successful propagation of the wave dynamics, as showcased by FC-LSTM (not even converging to a satisfying training error) and GridLSTM, which peaked in an excellent training error but failed in all test cases. This calls for an additional reasonable representation of space next to time.
- ConvLSTM gates a recurrently evolving latent state with convolution operations, conveniently combining spatial with temporal inductive biases. Accordingly, ConvLSTM successfully predicts the simple spatiotemporal dynamics into the future. On the more complex dataset, it depends on a comparably large number of parameters, though, which pays off in a limited generalization ability beyond the data distribution encountered during training, as discussed further below.
- While a direction-invariant configuration of DISTANA could propagate the simple and non-interacting wave dynamics, the introduction of eight direction-sensitive neurons—allowing the dedicated processing of lateral information by retaining the direction from where messages originate—substantially boosted the performance on the complex dataset.

In sum, the location-invariant sharing of weights, along with a direction awareness and temporally evolving latent states in recurrent systems (relating to sharing weights over time), guaranteed success in modeling two-dimensional spatiotemporal wave dynamics of various complexity. This aligns well with the concept of underlying causal principles, which are assumed to govern wave dynamics invariant to space and time. When satisfying this underlying principle in an appropriate model design, the spatiotemporal dynamics could be learned with low effort. Conveniently, the

location-invariant PK facilitated the simulation of wave dynamics in a spatial domain of arbitrary size.

Number of Parameters, Overfitting, and Generalization With 2.9 k parameters, *ConvLSTM8* reached outstandingly small errors on a regular test set (where the model was tested on similar conditions and data distributions as encountered during training). However, it performed poorly in a generalization task, when trained on a single wave sample and evaluated on a set of waves that were initialized in different locations of the grid, which is a clear sign for overfitting. The small *ConvLSTM1* (144 parameters), on the other hand, did not exhibit this weak generalization behavior, but did not achieve satisfying performance in general, while DISTANA robustly outperformed the baselines on all test sets (including generalization tests) without suffering from overfitting, not even when configured with many parameters.

Again, the design choice of DISTANA likely constrains the search space in a beneficial manner, where lateral information is propagated explicitly via a latent feature map that is decoupled from the actual system dynamics.[3] As a result, even the higher parameterized *DISTANA26* with 2.9 k parameters does not overfit to the training data, which manifests in a robust generalization ability.

Limitations The strong quantitative performance of the simple baselines indicate limitations of the MSE, depicting it as a non-ideal metric. More concretely, MSE yields large values if two sig-

3 This stands in stark contrast to ConvLSTM, which propagates lateral information via the observed dynamics by applying a 3×3 convolution directly on the input.

nals are identical and only shifted by a small amount in time.[4] Accounting for this shortcoming in the error measurement, we calculated the dynamic time warping (DTW) distance (Salvador and Chan, 2007) in a follow up study, which is not included here (Karlbauer *et al.*, 2020b). Even though DTW revealed a marginally different error pattern, we concluded that MSE in principle shows the same tendencies.

Crucially to mention, the lateral information propagation scheme in DISTANA introduces a latency in time, which can turn out as beneficial or disadvantageous, depending on the task to accomplish. That is, lateral information travels spatially between adjacent PKs, with an effective delay of one timestep. For example, PK_n sends a message about its current state in timestep t, which reaches PK_{n+1} in timestep $t+1$, where n is an index in the grid of PKs. In our experiments, we have not observed malicious behavior caused by the latency in lateral information flow. However, when increasing the wave propagation speed in the data generation process beyond a particular value, DISTANA was indeed no longer able to simulate the dynamics. This inability, however, is not a limitation of DISTANA per se, but relates to the Courant-Friedrich-Lewy (CFL) condition from computational numerics, where the propagation speed of the modeled quantity must not exceed one discretization unit (i.e., one pixel) in one simulation time step. Otherwise, the spatially propagating dynamics are technically impossible to be modeled.

Another shortcoming of this study lies in the sole consideration of synthetic data. Albeit using two datasets—where the second increased in complexity over the first—the investigations conducted

4 Consider two sine waves with identical amplitude and period, but slightly shifted in phase. Ideally, a metric would identify these two signals as fairly similar, returning a low error score. This is not the case for MSE, which is agnostic to positional shifts either in time or space.

in this RU do not provide insights about DISTANA's applicability to real-world dynamics. In the following section, we report our observations of applying DISTANA to global air temperature and pressure forecasting.

4.2 Research Unit II: Land-Sea-Mask Inference

4.2.1 Results

Experiment I: Static Context Inference in Shallow Water

Data Description In Dataset II, considered in Section 4.1.1 of RU I, the two-dimensional wave equation—also referred to as SWE—was solved to generate synthetic data. Here, we tweak one parameter of the SWE, we call it c, that accounts for water depth. Previously c was a scalar, which was constant over the entire 16×16 grid. We now define $\boldsymbol{c}$ as a two-dimensional tensor of size 16×16, taking different values per location. Effectively, $c_{i,j} \in (0, 1]$, with i and j as grid indices. While $c \to 0.0$ stands for infinite water depth and zero wave propagation speed, $c = 1.0$ encodes flat water, leading to high wave propagation velocities. An exemplary two-dimensional map of $\boldsymbol{c}$ is shown in Figure 4.3.

Gradient-Based Water Depth Inference Going beyond the sole simulation of two-dimensional wave dynamics, as conducted in the previous RU, we now recruit DISTANA in a knowledge

Figure 4.3: Inference of water depth when simulating the SWE. From left to right: Iteration 1, 2, and 500 of the inference procedure with inverted value range in $\boldsymbol{c}$ compared to the ground truth shown in the fourth panel. In the rightmost panel, the inference process of the water depth values is visualized. Figure modified from Karlbauer *et al.* (2021).

extraction task. That is, we seek to reveal hidden causes that parameterize an observed dynamic process, such as the water depth tensor $\boldsymbol{c}$, affecting the wave propagation speed in the SWE.

Technically, we simulate the wave dynamics—which exhibit different propagation speeds per location, depending on $c_{i,j}$—and project the observed error gradient reversely through the network onto a latent feature map $\hat{\boldsymbol{c}}$. While $\hat{\boldsymbol{c}}$ is initialized as a zero-tensor, it is iteratively shaped by the gradient signals, eventually resembling the topography of the true water depth map $\boldsymbol{c}$. When providing DISTANA with $\hat{\boldsymbol{c}}$ as additional input, the prediction accuracy increases substantially. Note that we do not treat $\hat{\boldsymbol{c}}$ as input to the PK's LSTM, but forward it to its gates only, as detailed in the following.

Advancing DISTANA with a Custom LSTM Cell We further increase the stability of DISTANA to model spatiotemporal wave dynamics over hundreds of autoregressive time steps.[5] This is achieved by routing the inferred context $\hat{\boldsymbol{c}}$ to the LSTM gates, but not to the cell input, and by removing any bias neurons from the system. Doing so, the activity in the PKs remains zero until the wave effectively reaches the location of the PK in the simulated grid, ultimately initiating the cell dynamics just when appropriate.

On balance, the inferred context map $\hat{\boldsymbol{c}}$ is realistic—even preserving a linear ordering of c-values that have not been encountered during training—and successfully parameterizes DISTANA to simulate the location-specific wave dynamics most accurately.

5 Due to space constraints in the conference publication, details about the modified LSTM in the PK are reported in an extended arXiv preprint, available under `https://arxiv.org/abs/2009.09823v1`

Experiment II: Predicting Global Air Pressure and Temperature

Data Description ERA5 (Hersbach *et al.*, 2020) is the fifth generation reanalysis dataset maintained by the ECMWF. It dates back to 1940 and covers a wide variety of Earth System variables, including states of the atmosphere, oceans, and soil. Based on ERA5, Rasp *et al.* (2020) created WeatherBench,[6] a selection of 14 key atmospheric variables on 13 vertical levels, including wind, pressure, humidity, temperature, solar forcing, as well as constants such as topography and LSM.[7]

Predicting Pressure and Temperature Progression To assess DISTANA's applicability to complex real-world dynamics, we train two separate models on predicting the WeatherBench variables Z_{500} and T_{850}, which translate to geopotential height on 500 hPa (a pressure field defined about 5.5 km above sea level) and air temperature on 850 hPa (roughly in 1.5 km altitude). We use time-series with a sequence length of 96, defined on hourly resolution, and employ the first 24 h for teacher forcing to tune the model into the current dynamics, and the remaining 72 h for autoregressive closed loop prediction, effectively unrolling a 3-days forecast into the future.

Yielding a RMSE of 816, we observe DISTANA to reasonably improve over the RMSE of 1114, produced by the competitive iterative DL approach on 5.625 ° degrees resolution reported in WeatherBench, when predicting Z_{500} for three days into the future. Admittedly, though, DISTANA is not unattained: Sophisticated state-of-the-art approaches from DL and NWP produce

6 `https://github.com/pangeo-data/WeatherBench`

7 The LSM is a two-dimensional tensor that encodes the ratio between land and water (oceans and lakes) that populates each pixel.

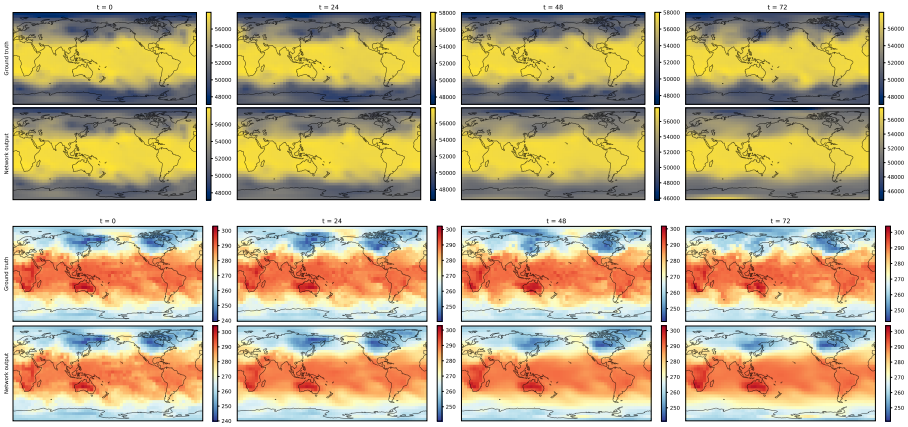


Figure 4.4: Geopotential height (first two rows with ground truth and network output) and air temperature prediction (bottom two rows) after 24 h, 48 h, and 72 h of autoregressive closed loop operation. Figure modified from Karlbauer *et al.* (2021).

RMSE scores of 268 and 154, respectively. Conceptually similar results are obtained when considering T_{850} for prediction. Irrespective of the forecast variable, DISTANA tends to create smooth fields when being unrolled into the future, thus progressing towards climatology, as visualized in Figure 4.4.

Static Context Inference to Support Forecast Accuracy Similarly to our experiment in Section 4.2.1, where water depth was inferred merely from observing the wave progression dynamics, we introduce a latent feature map $\hat{\boldsymbol{s}}$ on which we project DISTANA's errors when predicting Z_{500} and T_{850}. Resulting context maps $\hat{\boldsymbol{s}}$ are provided in Figure 4.5.

Despite the limited skill in DISTANA's forecasts, we make two intriguing observations, which are further discussed in the next section:

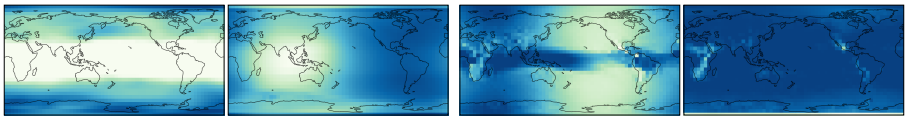


Figure 4.5: Final static context maps $\hat{\boldsymbol{s}}$ of four different model trainings, when projecting DISTANA's forecast errors of geopotential height (both left) or air temperature (both right) onto a latent feature map. Figure modified from Karlbauer *et al.* (2021).

- First, the final static context tensors $\hat{\boldsymbol{s}}$ that are shaped during training differ substantially, effectively depending on the forecast variable. More concretely, when predicting the comparably smooth geopotential height field, $\hat{\boldsymbol{s}}$ tends to encode latitude information, whereas notions of the LSM develop when predicting air temperature.
- Second, when enabling the gradient-based formation of $\hat{\boldsymbol{s}}$, the forecast error decreases slightly but reliably. Depending on the quality and usefulness of $\hat{\boldsymbol{s}}$, which varies over different model trainings, the forecast RMSE can be reduced by 5-10 %.

4.2.2 Discussion

In two experiments on synthetic and real-world data, we examined DISTANA's suitability as knowledge distillation method. Before recruiting the model for process explanation, though, it is required to simulate the observed dynamics sufficiently well. Thus, in the first experiment, we trained DISTANA to simulate the two-dimensional SWE with different wave-propagation speeds, varying in location as encoded in a context tensor $\boldsymbol{c}$ that was never provided to the model. During training, we projected prediction errors onto a latent feature map $\hat{\boldsymbol{c}}$ to account for location-specific param-

eterizations of the observed dynamics. In the second experiment, we explored DISTANA on predicting two variables contained in WeatherBench (Rasp *et al.*, 2020), namely Z_{500}, a pressure field at 500 hPa, and T_{850}, the air temperature at 850 hPa.

Iterative Pressure and Temperature Prediction The application of DISTANA to forecast Z_{500} or T_{850} improved over the naive iterative ML approach reported in WeatherBench, verifying its applicability to modeling real-world Earth System processes. Nevertheless, several implementation details—discussed in the *limitations* section below—prevented DISTANA from achieving the state-of-the-art results of sophisticated ML and NWP models.

Static Context Inference In Experiment I (detailed in Section 4.2.1), the two-dimensional water-depth tensor $\boldsymbol{c}$ has been inferred accurately into $\hat{\boldsymbol{c}}$ via a gradient-based error projection process applied throughout the training.[8] We thus conclude that DISTANA is well suited for revealing hidden relations in data, such as location-specific parameterizations of a dynamic spatiotemporal process.

Applying this procedure to real-world data of the atmosphere in Experiment II (see Section 4.2.1) resulted in the formation of intuitive static context maps $\hat{\boldsymbol{s}}$ that differed meaningfully when training the model to either predict air pressure Z_{500} or temperature T_{850}. The development of LSM nuances when predicting air temperature appears intuitive, since T_{850} is driven by the diurnal cycle of the land surface, heating up and cooling down immensely, whereas the diurnal temperature cycle is only marginally emphasized over wa-

8 Similar successes were obtained in later studies where we employed a physics-aware neural network to simulate the SWE and infer ICs, BCs, and water depth (Horuz *et al.*, 2022, 2023).

ter (Karlbauer *et al.*, 2024). In contrast, Z_{500} is hardly effected by the LSM, but exhibits patterns that structurally separate the polar regions from the midlatitudes, subtropics, and tropics. Accordingly, the formation of an almost discrete latitude-coding appears intuitive, as emphasized in the left-most map of Figure 4.5.

In balance, the inferred context maps $\hat{\boldsymbol{s}}$ supported DISTANA in predicting the respective dynamics into the future. This demonstrates the usefulness of the context maps, promoting them as parameterizations that could shed light on unknown relations between different variables in Earth System modeling. A reliable determination of such interrelations, however, would benefit meaningfully from an improved model that is able to simulate the observed real-world dynamics most accurately.

Limitations When comparing our approach to recent DLWP models, we identify several key differences: First, the hourly time resolution in our experiment is much finer than the typical time resolution of 6 h that finds application in competitive architectures (Scher and Messori, 2018; Weyn *et al.*, 2019, 2020, 2021; Pathak *et al.*, 2022; Lam *et al.*, 2022; Chen *et al.*, 2023). Interestingly, findings in Scher and Messori (2018); Bi *et al.* (2023) furthermore indicate finer time resolutions as more challenging for ML models, in particular when increasing the forecast lead time. Second, our model has a very limited field of view, where only direct neighbors are considered for prediction. Extending DISTANA's field of view, e.g., by introducing a hierarchy of model layers, most certainly will improve its predictions. Third, we did not account for data distortions in the polar regions, which originate from the equirectangular projection of the ERA5 data. When applying DISTANA to the distorted data, the location-invariant PKs are required to

learn a function that behaves differently depending on latitude.[9] While this is not trivial in the first place, it gives an explanation for the development of latitude codes in the inferred context maps $\hat{\boldsymbol{s}}$ irrespective of the predicted variable (refer to Figure 4.5).

As reported in Section 4.2.1, our model predictions become increasingly blurry during the successive autoregressive forward stepping. This might be a result from optimizing our model over a 96 h period. However, the predictability of the atmosphere decreases as a function of lead time (Somerville, 1987; Palmer and Hagedorn, 2006; Palmer, 2019). Accordingly, a model that is optimized on sequences beyond, say, 24 h, inevitably starts to increasingly generate average states of the atmosphere to best match its uncertainty, which eventually leads to climatology. This behavior—known as "regression to the mean" (Barnett *et al.*, 2005), where models tend to blur their predictions (Mathieu *et al.*, 2015)—could be prevented when introducing an explicit notion of uncertainty, such as realized in DL models for precipitation forecasting (Espeholt *et al.*, 2021; Ravuri *et al.*, 2021), as detailed in Thuemmel *et al.* (2023), or by constraining the optimization cycle to 24 h, as demonstrated by Weyn *et al.* (2019, 2020, 2021).

In general, the application of pure ML models bears the risk of non-plausible predictions. In Earth System modeling, certain physical rules are well established and numerical models are implemented to behave according to them. That is, they satisfy physical constraints, such as mass conservation or energy balances. Our DISTANA does not implement any explicit physical constraints and, accordingly, cannot be guaranteed to behave in a physically plausible way. In the next RU, we approach this limitation with the design of a physics-aware model that builds on DISTANA.

9 A latitude-dependent modulation of the PKs' dynamics could effectively be realized by an appropriate parameterization scheme. However, this would require the PKs to learn a function of reasonably higher complexity.

4.3 Research Unit III: FINN

4.3.1 Results

Experiment I: Simulating Synthetic PDEs

In a series of four benchmarks, we contrast our physics-aware FINN against pure ML models, namely TCN, convLSTM, and DISTANA, as well as against competitive physically inspired ML models, including Physics-Informed Neural Network (PINN) (Raissi *et al.*, 2019), CNN-NODE, which combines a CNN with the neural ordinary differential equation solver (Chen *et al.*, 2018), PhyDNet (Guen and Thome, 2020), Fourier neural operator (FNO) (Li *et al.*, 2020), and APHYNITY (Yin *et al.*, 2020).

For our benchmark, we choose the two-dimensional Burger's equation (Basdevant *et al.*, 1986), a one-dimensional diffusion-sorption equation (Nowak and Guthke, 2016), the two-dimensional Fitzhugh-Nagume equation (Klaasen and Troy, 1984) as a diffusion-reaction process (Turing, 1952), and the one-dimensional Allen-Cahn equation. While each of these PDEs has its own pitfalls—ranging from strong non-linearities and singularities to coupled equation systems with constituents of high degree—we focus our report on Burger's equation here. Results on the remaining three benchmarks underline and support the findings made with Burger's equation.

Dataset Description To assess the models' generalization capacity, we construct three different datasets from each PDE:

1. A *train* dataset consisting of a single sample of the respective PDE with an explicitly defined IC and BC. Pure ML models are trained on roughly the first 70 % time steps of the sequence and validated on the remaining 30 %.

2. A simple test dataset, called *in-dis-test*, that is simulated with the same IC and BC as used for the *train* dataset. Thus, the *in-dis-test* sample lies in the same distribution with the *train* dataset. Since the purpose of *in-dis-test* is to explore the models' temporal extrapolation ability, we generate a sequence that is twice as long as the *train* sequence. Models are thus required to simulate the dynamics far beyond the time horizon they were exposed to during training.

3. A more challenging test dataset, referred to as *out-dis-test*, with either different IC or replaced BC compared to the *train* dataset. With *out-dis-test*, we seek to validate the models' generalization ability beyond the statistics from the training period. To successfully simulate the *out-dis-test* sample, a model must be able to unroll the spatiotemporal process starting from an arbitrary IC and be robust to different behaviors of the simulation area's boundaries, i.e., handle various BCs.

Model Performance We first report the results of the pure ML models on Burger's equation and continue with the physics-aware approaches below. Errors in Table 4.3 are reported as relative mean squared error (rMSE), which is the MSE divided by the signal's variance.

Consistent with our findings from RU I, neither TCN nor ConvLSTM produced satisfying results, indicated by comparably large rMSE scores of $(3.1 \pm 0.8) \times 10^{-1}$ and $(6.0 \pm 6.7) \times 10^{0}$ on the *in-dis-test* sequence, or $(9.3 \pm 2.1) \times 10^{-2}$ and $(4.2 \pm 3.5) \times 10^{-1}$ on *out-dis-test*. Among the pure ML models, DISTANA produces the best result, reflected by error scores of $(2.9 \pm 4.3) \times 10^{-1}$ and $(3.6 \pm 3.6) \times 10^{-2}$ on the two test sets. These findings are supported qualitatively in Figure 4.6, where only DISTANA approxi-

Table 4.3: Comparison of relative MSE (rMSE, i.e., the MSE divided by the variance) and standard deviation scores across ten repetitions between different pure ML (above dashed line) and physics-aware neural networks (below dashed line) on Burger's equation. Best results reported in bold. Table adapted from Karlbauer *et al.* (2022).

		Dataset		
Model	(Params)	*Train*	*In-dis-test*	*Out-dis-test*
TCN	(29 690)	$(3.9 \pm 1.3) \times 10^{-2}$	$(3.1 \pm 0.8) \times 10^{-1}$	$(9.3 \pm 2.1) \times 10^{-2}$
ConvLSTM	(22 600)	$(2.4 \pm 3.7) \times 10^{-1}$	$(6.0 \pm 6.7) \times 10^{0}$	$(4.2 \pm 3.5) \times 10^{-1}$
DISTANA	(19 100)	$(7.3 \pm 11.0) \times 10^{-3}$	$(2.9 \pm 4.3) \times 10^{-1}$	$(3.6 \pm 3.6) \times 10^{-2}$
PINN	(3 041)	$(1.8 \pm 0.8) \times 10^{-1}$	$(5.8 \pm 2.5) \times 10^{-1}$	—
CNN-NODE	(2 657)	$(2.0 \pm 0.7) \times 10^{-3}$	$(4.7 \pm 6.5) \times 10^{-1}$	$(9.1 \pm 9.5) \times 10^{-1}$
PhyDNet	(185 300)	$(1.1 \pm 0.3) \times 10^{-3}$	$(2.6 \pm 2.4) \times 10^{-1}$	$(3.5 \pm 1.5) \times 10^{-1}$
FNO	(116 115)	$(6.4 \pm 9.2) \times 10^{-4}$	$(5.9 \pm 2.6) \times 10^{-2}$	$(9.7 \pm 0.9) \times 10^{-1}$
APHYNITY	(2 658)	$(1.0 \pm 0.3) \times 10^{-2}$	$(4.3 \pm 0.1) \times 10^{-2}$	$(4.4 \pm 0.5) \times 10^{-4}$
FINN	(**421**)	$\mathbf{(1.0 \pm 0.6) \times 10^{-4}}$	$\mathbf{(1.1 \pm 0.4) \times 10^{-2}}$	$\mathbf{(2.4 \pm 1.0) \times 10^{-5}}$

mates the ground truth with sufficient detail.

On the other end, even physics-inspired approaches have difficulty predicting the dynamics of Burger's equation into the future. None of PINN,[10] CNN-NODE, and PhyDNet reaches satisfying errors on the *in-dis-test* and *out-dis-test* samples (cf. Figure 4.6). While FNO is competitive on the *in-dist-test* sample, it shows no generalization ability on *out-dist-test*. Intriguingly, both APHYNITY and FINN solve the two tasks by generating accurate approximations of the *in-dis-test* and *out-dis-test* samples. Even though FINN counts an order of magnitude less parameters than APHYNITY, it produces the smallest rMSE scores, outperforming APHYNITY by an order of magnitude on *out-dis-test*.

10 Due to the explicit formulation of the PDE of interest when constructing PINN, including IC and BC, the model cannot, by design, be applied to conditions other than those it has been trained on.

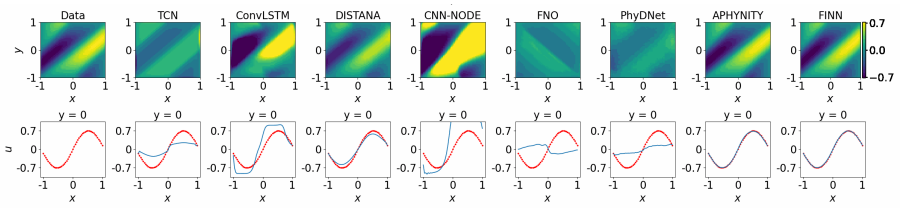


Figure 4.6: Burgers' data (red) and model predictions (blue) of the *out-dis-test* sample. First row shows the solution over x and y at the end of the sequence, and the second row visualizes the according solution and prediction over x as a cross-section at $y = 0$. Credit for creating this figure goes to Timothy Praditia.

Knowledge Distillation A unique property of FINN lies in the compositional model design, which allows the formulation of a desired component of the considered PDE as an ANN. In each of the chosen benchmarks, we therefore test FINN's ability to reveal the function of a dedicated component in the equation. When simulating Burger's equation with FINN, we define the advection velocity $v(u)$, which depends on the quantity u that is target for prediction, as a learnable component in form of a five-layered MLP with [1, 10, 20, 10, 1] neurons, respectively. Furthermore, we set the diffusion coefficient $\mathcal{D}$ as a learnable parameter.

Excitingly, the advection velocity function $v(u)$, formed by the small MLP component in FINN during training, precisely resembles the true advection velocity function, which was used when generating the samples. Similarly, FINN reliably reveals retardation factor functions in the diffusion-sorption and Allen-Cahn benchmarks, as well as two-dimensional reaction functions when modeling the diffusion-reaction equation.

Experiment II: Modeling Contamination Transport in Saturated Clay

Dataset Description In Nowak (2000), three core samples were taken from an area where a contaminant was spilled over unsealed ground, which has a non-diffusive layer of clay at its bottom. To simulate the distribution of the contaminant concentration u over time through the soil, the three samples were experimentally spilled with the contaminant from top while recording the resulting concentration at the bottom of the samples, producing a sequences of 55 time steps showing the concentration of u over time. For this real-world dataset, no retardation factor function $R(u)$ is known. The goal of this experiment, thus, is to employ FINN to determine $R(u)$ purely from data.

Model Performance Despite the very sparse training sequence (only 55 time steps), FINN learns to simulate the diffusion of the contaminant through the soil with high accuracy, as sketched in Figure 4.7. With an rMSE of 8.28×10^{-3}, FINN even outperforms a calibrated physical model, which produces an rMSE of 1.52×10^{-2}. This result is supported by an accurate representation of the contamination concentration profile in core sample #2B at the end of the experiment, where FINN and the calibrated physical model achieve rMSE scores of 6.39×10^{-2} and 1.50×10^{-1}, respectively.

Most importantly, the retardation factor function $R(u)$ learned by FINN can now be extracted from the MLP, as visualized in Figure 4.7. In fact, the non-linear shape of the learned $R(u)$ relates to that of a typical retardation factor function. That is, the diffusion rate of the quantity u is delayed strongest for small concentrations of u, followed by a steep decay of the retardation rate until a certain concentration is reached (here at a quantity of $u \approx 200$). This translates to slow exchange rates of u if the quantity of u is low

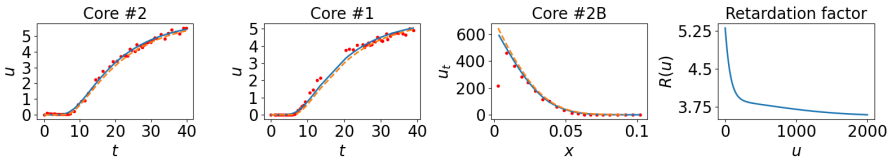


Figure 4.7: Breakthrough curve prediction of FINN (blue line) during training using data from core sample #2 (left), during testing using data from core sample #1 (second from left) and total concentration profile prediction using data from core sample #2B (second from right). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The right-most plot shows the learned retardation factor $R(u)$. Figure and caption text taken from Karlbauer *et al.* (2022).

and to high exchange rates once a critical concentration level is reached.

4.3.2 Discussion

In this RU, we introduced FINN, an ML model arranging different ANN components in a compositional manner, inspired from physics and its predecessor DISTANA, to adequately model advection-diffusion processes formulated as PDEs.

Accurate PDE Simulation When comparing FINN to several pure ML and other physics-aware ML models in numerous benchmarks on synthetic PDEs as well as on real-world data, we observed superior performance—often yielding errors that are orders of magnitudes smaller than those of the second best model—while requiring only a small fraction of trainable parameters.

While other models tended to overfit to the training sample, as observed for TCN, ConvLSTM, PINN (by design), CNN-NODE,

PhyDNet, and FNO, FINN demonstrated robust generalization abilities both in time (extrapolation) and when confronted with unseen ICs and new BCs. Only APHYNITY (with 2 657 parameters) produced results that were comparable to FINN, albeit depending on the knowledge of the complete PDE, whereas FINN (with as few as 421 parameters) can be applied to processes that are only partially understood by setting the unknown constituents of the considered PDE as learnable components.

Ultimately, the small number of parameters that are required to accurately simulate a wide variety of ICs and BCs with FINN confirms the effective architecture design. Note, however, that the components in FINN must be composed and connected adequately, forming appropriate inductive biases, to model the respective process of interest successfully.

Effective Inference of Unknown Relations Both in the synthetic and real-world datasets, FINN was conceptualized to learn an unknown component merely from data in order to simulate a spatiotemporal process of interest. FINN accurately determined the prevailed constituents in synthetic experiments, such as the advection velocity $v(u)$ in Burger's equation, and demonstrated competent process explanation capacities by developing a realistic retardation factor function in a real-world clay experiment, giving rise to determining soil properties even with very limited amounts of training data, i.e., only 55 data points in our training sequence. The inferred $R(u)$, for example, provides a concrete function of the contaminant's diffusion speed through soil, which allows an estimate of how fast the contaminant u will diffuse throughout the soil for what concentrations of u.

In follow-up studies, we explored FINN's flexibility to learn different types of sorption isotherms under consideration of prediction

uncertainties (Praditia *et al.*, 2022) and to infer the applied BC driving the observed process (Horuz *et al.*, 2022, 2023). Moreover, in unpublished work,[11] we extended FINN to model the SWE and replicated our earlier findings from Karlbauer *et al.* (2021) where the water depth was inferred during training, legitimating FINN as a tool to determine a static context tensor $\hat{\boldsymbol{s}}$ that parameterizes the observed process location-wise.

We conclude that FINN denotes an effective model for simulating spatiotemporal processes that evolve according to specific PDEs from the family of advection-diffusion processes. Note that the primary goal of this RU was not to construct a surrogate model to accelerate process simulation—such as developed in, e.g., Kochkov *et al.* (2021)—but to design a model for knowledge distillation, revealing unknown relations and interdependencies merely from sparse data. FINN proved robust against overfitting and demonstrated reliable generalization abilities with few learnable parameters. Conveniently, only limited understanding of the process that is subject for prediction is needed from the modeler when applying FINN. Thus, we promote FINN as a tool for accelerating and simplifying the simulation of substance propagation through a medium by affording less expert knowledge and experimentation, even in contexts where key aspects of the target process are not understood. Moreover, FINN can be employed with low effort to reveal hidden relations that parameterize the temporal evolution of spatiotemporal processes.

In cooperation with the cluster of excellence "Machine Learning: New Perspectives for Science" at the University of Tübingen, we distilled the content of this research project, resulting in a blog

11 Credits go to Coşku Can Horuz, who conducted the project as a part of his master's thesis under joint supervision of Sebastian Otte and Matthias Karlbauer.

entry available in English[12] and German[13] language to convey our core findings to a broader public audience.

Limitations While FINN implements only low barriers (compared to conventional numerical methods) for a modeler with limited domain knowledge who aims at simulating complicated PDE processes accurately, a certain degree of expertise in numerical simulation—such as a basic understanding of the finite volume method (Moukalled *et al.*, 2016) and the resulting discretization and information propagation scheme—remains a requirement in order to handle, develop, and calibrate FINN seamlessly.

Moreover, in its current formulation, FINN is only applicable to processes that can be described by first-order temporal, and up to second-order spatial derivatives. Thus, depending on the process that is subject for prediction, the model might require a substantial redesign. For example, when approaching the simulation of weather dynamics, the Navier-Stokes equations must be satisfied. Concretely, the PDEs describing the weather require the conservation of four quantities: momentum (Newton's 2nd law), mass (continuity equation), energy (thermodynamics 1st law), and water mass (moisture equation). Formulating a FINN-style model to simulate weather dynamics would correspond to the effort required to implementing a complete Earth System model. Thus, in the next RU, we shift gears back to a pure ML approach, albeit with physically inspired choices concerning data selection and preprocessing, forward stepping scheme, and architecture design.

12 `https://www.machinelearningforscience.de/en/fusing-physical-knowledge-with-neural-networks-flexibility/`

13 `https://www.machinelearningforscience.de/so-verschmelzen-wir-physikalisches-wissen-mit-der-flexibilitaet-neuronaler-netze/`

4.4 Research Unit IV: Competitive DLWP

4.4.1 Results

Data Description Accounting for the experiences of global air temperature and pressure forecasting made in Experiment II of RU II, we build on the hierarchical U-Net from Weyn *et al.* (2021) and advance their cubed sphere projection of the ERA5 data (described in Section 4.2.1) by recruiting the HEALPix mesh. Albeit the cubed sphere, which subdivides the sphere into six faces, reduces the distortions of the original ERA5 data's equirectangular projection already meaningfully, its discretized patches are not lined up on constant lines of latitude, which results in different orientations of individual patches. In contrast, on the HEALPix mesh, which separates the sphere into twelve faces, patches have an isolatitudinal arrangement, leading to a consistent "east to the right" orientation of all patches, which facilitates the development of location invariant convolution kernels that can be applied seamlessly in any latitude without the need for further parameterizations.

From the ERA5 dataset, we select seven prognostic variables, which are subject for prediction, including the four geopotential height fields Z_{1000}, Z_{500}, Z_{250}, and $\tau_{700-300} = Z_{700} - Z_{300}$, denoting the thickness between the Z_{700} and Z_{300} fields, two air temperature fields T_{850} and T_{2m}, and the total column water vapour ($TCWV$), which integrates the water vapour above one patch vertically into the atmosphere. Moreover, three prescribed inputs are prepared: The static topography map Z, the static LSM, and the $TISR$, which is a dynamic field that acts as solar forcing, driving our

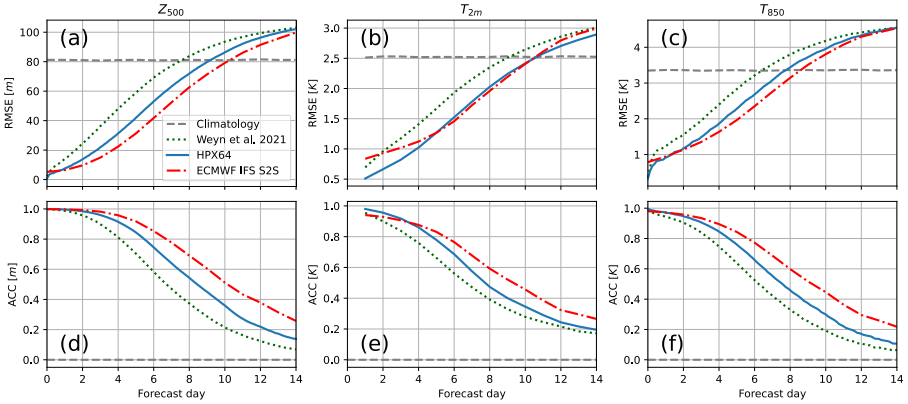


Figure 4.8: Performance comparison of our DLWP-HPX, Weyn *et al.* (2021)'s cubed sphere model, and ECMWF's IFS model, averaged over 208 forecasts. RMSE for (a) Z_{500}, (b) T_{2m} and (c) T_{850}; climatology is indicated by the gray dashed line. ACC for or (d) Z_{500}, (e) T_{2m} and (f) T_{850}. Figure and caption text modified from Karlbauer *et al.* (2024).

planet's atmosphere.[14]

Enhanced Weather Forecasts with Two Weeks Lead Time
We compare our DLWP model on the HEALPix mesh (DLWP-HPX) against its predecessor from Weyn *et al.* (2021) on the cubed sphere,[15], and against the state-of-the-art NWP model provided by ECMWF's IFS. The RMSE and ACC evolution of the three models over a fourteen days period is plotted in Figure 4.8.

14 $TISR$ can be precomputed from the relative positioning of Earth and Sun, time of day, and latitude and specifies the amount of energy that enters each patch of the discretized Earth at a specific date and time.

15 More details along with scores on the RMSE and anomaly correlation coefficient (ACC) metrics, resulting from successive improvements to the U-Net architecture, data choices, and forward scheme employed in Weyn *et al.* (2021) are reported in Karlbauer *et al.* (2024).

When predicting Z_{500} over five days into the future, the models produce RMSE scores of 56.01 (Weyn *et al.*, 2021), 41.91 (DLWP-HPX), and 31.30 (IFS). Specifically, our model gains two days against its predecessor and falls behind the state-of-the-art by only one day. The improvement over Weyn *et al.* (2021)—mostly resulting from the conversion to the HEALPix mesh, from inverting the channel depth in the U-Net, and from incorporating GRU cells—are also manifested in other prognostic variables, such as T_{2m} and T_{850}. Excitingly, our model seems to match the state-of-the-art on T_{2m}. However, the performance of the IFS model must be interpreted with caution: The imprecisions in the re-gridding of the IFS forecasts onto the $1\,^\circ \times 1\,^\circ$ mesh that we use for error computation, likely introduces errors to the representation of coastlines, which substantially influence the error computation of the temperature fields due to their susceptibility to the underlying surface (being land or water).

Stable Autoregressive Rollouts Over 365 Days While being only optimized on a 24 h cycle during training, our model can be operated autoregressively at test time for hundreds of simulation steps. In an exemplary rollout over 365 days, our model generates stable and plausible realizations of the atmosphere that conserve seasonal trends, as visualized in Figure 4.9 (a).

To test our hypothesis that the seasonal cycle is driven by the $TISR$ input by telling the model the amount of energy that enters the system at what location and time, we train a model without $TISR$ input and generate another 365-days rollout, plotted in Figure 4.9 (b). Although the model still manages to progress the boreal summer dynamics into a perpetual autumn (depicted by lower values of Z_{500} in the northern hemisphere), it fails at raising the energy level again when approaching the next year's spring and summer in the 365-days simulation. This confirms $TISR$ as

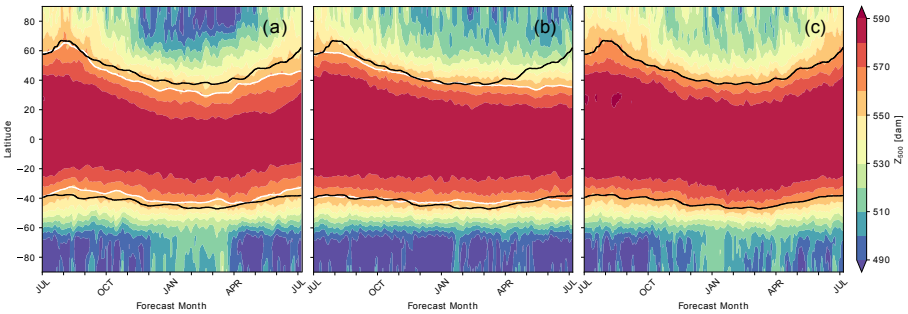


Figure 4.9: Zonally averaged three-day mean of Z_{500} plotted as a function of time and latitude: (a) for a recursive one-year model simulation initialized on 1 July 2017, (b) same as (a) for a model trained and tested without $TISR$, (c) the corresponding averaged Z_{500} field from the ERA5 reanalysis as ground truth. 15-day averaged values for the 560 dam contour for the ERA5 data (black line in all three panels) and for the DLWP-HPX simulation (white line in (a) and (b)). Figure and caption text modified from Karlbauer *et al.* (2024).

a key factor for generating seasonal trends by acting as external solar forcing, which the model can adhere to and consider when generating its predictions.

Moreover, in a spectral power analysis of the Z_{500} prediction along a constant line of latitude (45 ° N) at different lead times, we observe the model to loose some power until two days after initialization. Beyond two days, however, the power of the predicted atmospheric states excitingly remains stable without decaying any further, which supports our findings from the 365-days rollouts that do not wash out in form of blurred predictions, but reliably conserve high frequencies in space.

4.4.2 Discussion

We proposed a parsimonious U-Net to forecast seven global atmospheric states by advancing the model from Weyn *et al.* (2021). Most significant improvements resulted from replacing the cubed sphere by the HEALPix mesh, inverting the channel depth of the U-Net, and incorporating GRUs.

Albeit not quite on par with the state-of-the-art weather forecasts generated by ECMWF's IFS, falling behind by one day at a seven days lead time, our DLWP-HPX model produces competitive forecasts, especially when comparing the number of prognostic variables (7 in DLWP-HPX vs. 820 in IFS), the spatial resolution (1 ° vs. 0.1 °), and the temporal step size (3 h vs. 12 min).

While recent sophisticated DLWP, such as Pangu-Weather (Bi *et al.*, 2023) and GraphCast (Lam *et al.*, 2022), reach the skill of ECMWF's NWP forecasts, they suffer from shortcomings (e.g., instable behavior when generating forecasts beyond two weeks), which our model does not exhibit. We discuss the advantages of our model in the following.

Physically Plausible Predictions Even though we did not implement explicit physical rules according to which quantity is conserved, propagated, or parameterized, we have made various design choices based on physical reasoning. As postulated in Thuemmel *et al.* (2023), these physically inspired design choices have the potential to push the model's predictions into a physically plausible regime, as observed in our model, which exhibits quite realistic and stable dynamics. We detail our design choices that are inspired by physics in the following:

(1) **HEALPix:** According to our first attempts of global pressure and temperature prediction in RU II, we prevent our

Figure 4.10: Coastlines of our planet in the equirectangular Mercator projection (left), on the sphere (center), and in the distortion-reducing HEALPix projection (right). On a distortion-free representation, Greenland should fit roughly seven times into South America, which applies to the HEALPix, but not to the equirectangular representation. Color codes indicate how HEALPix divides the sphere into twelve faces: Four each on the northern and southern hemisphere (blue), and another four around the equator (orange).

model here from having to correct for data distortions, by projecting the ERA5 data to the HEALPix mesh before training our DLWP-HPX model to predict their temporal progression. As visualized in Figure 4.10, the HEALPix mesh substantially reduces the distortions of the equirectangular representation. Moreover, compared to the cubed sphere mesh, HEALPix implements an isolatitudinal alignment of patches, resulting in a consistent "east to the right" orientation of all patches in the mesh regardless of latitude, as emphasized in Figure 4.11. The resulting mesh closely matches the situation on the sphere and provides a physically sound context for training location invariant convolution kernels.[16]

(2) **Delta Prediction:** Solving PDEs so simulate the temporal dynamics of a system typically results in predicting the rate of change with respect to the current state, compared to predicting the absolute value in the next time step. This

16 Note that Weyn *et al.* (2021) trained two distinct sets of weights for their U-Net to separately learn to simulate the dynamics on the two polar faces and on the four equatorial faces of the cubed sphere mesh.

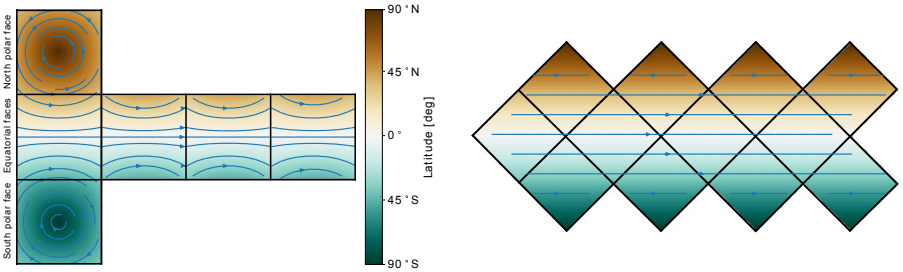


Figure 4.11: Lines of latitudes depicted as blue streamline arrows on the cubed sphere (left) and on the HEALPix (right). While the lines corresponding to constant eastward motion describe arcs of different radii on the cubed sphere mesh, the same motion translates to straight lines on the HEALPix mesh. Figure and caption modified from Karlbauer *et al.* (2024).

principle is well established in ML by employing residual connections, as proposed in the ResNet architecture (He *et al.*, 2016), and finds application in our model.

(3) **Paired Input:** Instead of providing our model with a single state per simulation step, we feed two consecutive states as pair into the network. This allows the model to determine first-order temporal derivatives between these two states, giving rise to considering the rate of change of each prognostic variable in addition to their absolute values in both states.

(4) **Parsimonious U-Net:** In line with the success of Pangu-Weather and GraphCast, which are sophisticated DLWP models based on the vision transformer (ViT) (Dosovitskiy *et al.*, 2020) and GNN (Battaglia *et al.*, 2018), we base our model on the parsimonious U-Net (Ronneberger *et al.*, 2015) in accordance with Tobler's first law of geography "All things are related, but nearby things are more related than distant things." (Tobler, 1970). In contrast to ViTs, the convolu-

tional structure of the U-Net formulates an implicit bias on the close neighborhood surrounding each patch.

(5) **Satisfying the CFL Condition:** Through the employment of dilations and via the hierarchical structure of the U-Net, we guarantee that the receptive field is large enough to cover the fastest moving particles governed by Jetstreams with velocities of up to 100 m/s. At an effective time step size of 6 h (two 3 h states in one prediction step), the fastest traveling signals propagate 2160 km/h in one simulation step. Our model operates on a mesh with an effective grid spacing of approximately 110 km and spans 175 patches, which amounts to $(175 \cdot 110\,\text{km})/2 = 9625\,\text{km}$.[17] Thus, our model's receptive field is large enough to cover the fastest moving signals in the atmosphere, which satisfies the CFL condition.

(6) **3 h Time Resolution:** Short-lived microscale phenomena in the atmosphere—such as heat transfer, turbulence, and convection, leading to the formation of clouds and to precipitation—evolve on small scales (below 1 km) and within short time windows. Thus, refining the spatial and temporal resolution seems beneficial for simulating the progression of the atmosphere.

Both Pangu-Weather and GraphCast, however, report larger errors when applying shorter time deltas. Instead, our model improved when switching from 6 h to 3 h time steps—albeit only after the incorporation of GRUs into the model pipeline—which might indicate that DLWP-HPX denotes a step into the direction of a DLWP core that adheres to principles from numerical simulation and physics.

In contrast to NWP, however, the refinement of the time step

17 Dividing by two gives the distance into one direction of the considered patch's center. Otherwise, the width of the entire kernel is obtained.

> does not seem to be a strict requirement for DLWP models that still resolve microscale processes successfully, as demonstrated by several competitive DLWP architectures (Weyn *et al.*, 2021; Pathak *et al.*, 2022; Lam *et al.*, 2022).

The above design choices potentially support our model in generating physically plausible forecasts when compared to state-of-the-art DLWP models. That is, instabilities in Pangu-Weather and GraphCast constrain their forecasts to less than two weeks lead time, while our DLWP-HPX model can be driven autoregressively to generate realistic forecasts over an entire year without approaching climatology and while preserving a high granularity of the predicted fields. We attribute this stability of our model to the comparably short optimization cycle of 24 h, as suggested and motivated in RU II (see *Limitations* in Section 4.2.2).

Another physically sound behavior of DLWP-HPX is suggested by the results in the 365-days rollouts, where a solar forcing input (i.e., $TISR$) improved the rendering of seasonal trends. Apparently, $TISR$ helped the model determine the season at each latitude and accordingly facilitated the simulation of warmer or colder climates, shown here as higher and lower pressure values in the Z_{500} field. Importantly, once $TISR$ is withheld from the model, it does no longer express seasonal trends, neither showing austrial summer in January, nor returning back to boreal summer in May to July ten months into the simulation, which reflects the effective physically plausible use of $TISR$ when available. This is emphasized by the white line on the northern hemisphere in Figure 4.9 (b), which does not follow the verification drawn in black. Noteworthy, though, the model still realized a stable forecast—even without a notion of seasons due to the missing $TISR$ input—when being operated in closed loop over several hundreds of simulation steps.

Experimentally shifting $TISR$ by several hours, such that it does not align with the time step of all other variables, dramatically

deteriorated the prediction of T_{2m}, which further demonstrated the effect of $TISR$ on the prediction of surface variables and showcased the model's sensibility to wrong solar forcing inputs. In contrast, though, Z_{500} was hardly effected by the manipulated $TISR$ input, which is reasonable since geopotential height is only marginally effected by direct incoming solar radiation.

Recurrence Technically, we employed ConvGRU (Ballas *et al.*, 2015) with a kernel size of $k = 1$[18]—as inspired by our experiences when developing DISTANA in RU I—enabling our model to evolve a latent state over time, which proved helpful in RUs I and II.

Experimenting with convLSTM and DISTANA did not improve over convGRU, which we attribute to the short time scale used for training our model (i.e., the 24 h optimization cycle reported above, effectively leading to four time steps). On such short time scales, the effect of a forget mechanism might be negligible. Further research could address DISTANA's stabilization and generalization benefits in the context of DLWP.

The improvements on the RMSE and ACC metrics, on the expression of seasonal trends, and on the conservation of high frequencies in long rollouts, resulting from the incorporation of recurrent connections, indicate that temporal processing and the development of a latent state over time contributes significantly to predicting the progression of atmospheric states. Investigating this further could reveal long-range dependencies that are unknown today, in particular since state-of-the-art NWP and DLWP models do not implement capacities to memorize previous states of the system.

18 Note that ConvGRU with $k = 1$ resembles a GRU that is cast along the spatial dimensions of a feature map without considering neighboring positions.

Limitations Even though DLWP-HPX was demonstrated to generate physically plausible predictions, it is not guaranteed to satisfy conservation laws or to respect energy balances. Such physical constraints could, most conveniently, be implemented as part of the loss function to, e.g., enforce divergence as demonstrated by De Bézenac *et al.* (2019). Optionally, the model architecture can be designed carefully to adhere to physical laws, such as our FINN, developed in RU III (see Section 4.3) More alternatives to incorporate physical constraints into ML models have been discussed in Section 3.2.2.

Since our model was trained over a 24 h cycle only, the recurrent latent states h could only evolve over four time steps, which is considered very few in time-series forecasting. This prevents the model from developing a stable recurrent attractor that encodes the currently ongoing dynamics, parameterizing the subsequent convolutions accordingly and driving the forecast under consideration of previous states. In fact, to prevent h from exploding when operating the model autoregressively in closed loop during inference, we had to reset h after each 24 h cycle. As a consequence, the model has no capacity to capture effects that exceed a temporal latency of one day.

Chapter 5

Conclusion and Outlook

In this thesis, we travelled through the simulation of spatiotemporal processes, ranging from synthetic wave dynamics to real-world weather mechanics, by employing ML models under consideration of physical knowledge—either explicitly enforced or implicitly considered via dedicated architecture design. The overarching goal of this thesis was to determine key design elements for the simulation of spatiotemporal processes and to explore and apply techniques for knowledge extraction from ML models to advance process understanding in real-world systems.

In four dedicated RUs, each featuring one publication included in Appendix A, we investigated four research questions, detailed in Section 3.4. The results of each RU are reported and discussed in Section 4.1, Section 4.2, Section 4.3, and Section 4.4, respectively. We proceed with an overall discussion that involves concluding remarks below, followed by an outlook on promising future research directions.

5.1 Conclusion

5.1.1 Physics-Inspired Model Design

An overarching topic among all RUs constitutes the model design for spatiotemporal processes. While model architectures in RUs I, II, and IV were implicitly inspired by physical principles, such as location and time invariance when simulating a spatiotemporal process, cf. Thuemmel *et al.* (2023), RU III implemented an architecture that explicitly satisfies physical processes, such as advection and diffusion, describing substance transport through a medium.

Taken together, the results from RUs I through IV well-support the benefit of physics-inspired model design. When contrasting our models against architectures with less physical priors, we observed both faster convergence, and better performance (even with less parameters) when learning to simulate spatiotemporal dynamics. Moreover, in our experiments throughout all RUs, we observed benefits of models with recurrent connections over models without such connections. Even though recurrent models are considered more challenging to train, they allow the evolution of a latent state that keeps track of past information that is not necessarily observable in instantaneous system states. Appealingly, recurrent connections satisfy the notion of temporal invariance by learning a single function to iteratively project dynamics into the future.

Most prominently, however, in RU III, we found that explicit physical design choices, implemented as inductive biases (Battaglia *et al.*, 2018), have the potential to outperform other models by a large margin. Carefully designed physics-aware ANNs[1] can learn physical processes even more accurately, while requiring just a frac-

1 The implementation of inappropriate inductive biases most likely prevent models from adequately learning processes that adhere to rules that do not align with the chosen inductive biases.

tion of the data required by pure ML models, which is consistent with observations from Karniadakis *et al.* (2021). To successfully build a physics-aware ANN, however, the designer must be well aware of certain physical principles from numerical simulation.

Thus, when a system is known to be governed by certain (physical) principles, it is highly beneficial to constrain a DL model accordingly, particularly when facing out of distribution applications. Such constraints might either be reflected by implicit model design, or via explicit formulations that enforce the model to obey to given rules. Nevertheless, as demonstrated in RU IV, implicit model design can lead to models that adhere well to physical principles, even in complex and potentially chaotic systems such as our planet's atmosphere. Lastly, when it comes to climate projections, where models are operated autoregressively for extremely long lead-times, likely reaching out of distribution domains, the incorporation of physical constraints is expected to be of value to guarantee the model predictions to stay in a physically plausible regime. The role and necessity of explicit physical constraints, however, remains open for debate and exploration.

Indeed, with the recent advent of applying ML in ESM modeling, various authors have outlined the potential of combining ANN's learning ability with domain knowledge from physics (Bauer *et al.*, 2023; Tesch *et al.*, 2023). More specifically, Shen *et al.* (2023) emphasize the differentiability of DL models, which paves the way for sophisticated model analyses that reveal input-to-output relations and showcase the relevance of individual input variables.

5.1.2 Knowledge Gain

In RU III, we augmented diffusion-advection equations with ANN modules to learn unknown relationships from data, such as reaction terms or advection velocities. This depicts only one way of success-

fully applying DL for revealing unknown functional relationships from data. Alternatively, eXtreme Gradient Boosting (XGBoost) or input permutation methods (Breiman, 2001) can be applied with MLPs to reveal the relevance of input variables. We employed these techniques to reveal unexpected relationships when predicting heat flux, where solar input radiation turned out as a key driving variable, although it has never been considered in according physical models before (Wulfmeyer *et al.*, 2023).

A third method to extract knowledge from ANNs exploits the differentiability of DL models and projects gradients onto model inputs. This procedure can be used to clean noisy observations (Otte *et al.*, 2020) or to reveal hidden causes that affect the evolution of spatiotemporal processes, as demonstrated in RU II, where a global LSM was inferred merely from observing and predicting global temperature dynamics.

In balance, throughout all experiments, we observe enormous potential in clever model design, which complements common knowledge about scaling laws in DL.[2] While the application of such scaling laws requires ever increasing compute facilities, this does not necessarily hold for dedicated architecture designs, tailored to specific tasks. Particularly, we experience better performance with less parameters and data when designing DL architectures carefully. It remains an open question, though, whether specific DL model designs can lead to foundation models (FMs) that can solve arbitrary tasks (Bommasani *et al.*, 2021). Casting an eye on biology and the way how biological agents learn when exploring their environment, however, we should expect that energy efficient and extremely sophisticated and dedicated model architectures, given to any living agent, are indeed able to solve a vast variety of tasks.

2 Scaling laws prove the value of increasing model parameters (Kaplan *et al.*, 2020) or the amount of training data (Hoffmann *et al.*, 2022) and diminish the role of architecture design.

5.2 Outlook

The results and observations made in the four RUs of this thesis open avenues for research directions to advance spatiotemporal modeling, weather forecasting, and knowledge gain through the application of machine learning models. In this respect, various open topics that remain for exploration are outlined in the following.

Refining Temporal Resolution Most DLWP models operate on a six-hour time delta, which proved to be a reasonable trade-off between training complexity[3] and granularity, that is, spatial resolution, of the model output. From the theory of NWP, smaller time deltas should facilitate more accurate predictions, which has not been observed in DLWP so far, though. In contrast, state-of-the-art DLWP models tend to produce worse predictions under finer temporal resolutions.

Nevertheless, in RU IV we have demonstrated that our DLWP-HPX model in fact benefits from a smaller time delta, i.e., 3 h resolution, however, only in combination with recurrent connections. Further research is needed to pinpoint the relationship of temporal resolution and forecast accuracy in DLWP models.

A promising direction of refined temporal resolution lies in the implementation of multiple temporal resolutions on different resolutions in space. That is, simulating weather dynamics on multiple spatial hierarchies with appropriate time stepping sizes per hierarchy, satisfying the CFL condition. This has been proposed by Rangapuram *et al.* (2023), albeit on pointwise time-series forecasting problems only and not on spatially distributed data.

Such multiple temporal hierarchies might also be tailored to differ-

3 Smaller time deltas require more autoregressive steps to generate, say, a two-weeks forecast, quickly resulting in diverging activities.

ent prognostic variables—sea-surface temperature (SST) dynamics, for example, evolve on much slower temporal scales than air temperature dynamics, i.e., daily versus hourly—and could further be combined with predictors on each level of the hierarchies. This has been proposed in Lin *et al.* (2017), where the predictions at each level contribute to the overall loss.

Developing an ESM Coupling the atmosphere with Earth surface dynamics is an essential component of NWP and constitutes an indispensable ingredient for holistic weather forecasting. On land, an ESM can implement the consideration of surface heterogeneity (Bou-Zeid *et al.*, 2020) and vegetation (Cowan *et al.*, 1977; Schymanski *et al.*, 2008; Warrach-Sagi *et al.*, 2022) to consider, e.g., evapotranspiration and the feedback of the surface on the atmosphere. Over sea, the interaction of ocean and atmosphere is an established research field (Kraus and Businger, 1994), which demonstrated the effect of the ocean's SST on the evolution of winds, potentially leading to tropical cyclones, i.e., hurricanes and typhoons (Liu and Alexander, 2007; Chelton and Xie, 2010). This bears great potential for DLWP, which has not yet been formulated to combine the prediction of states in both the atmosphere and the ocean in a unified model.

Nested DLWP To reduce computational complexity, nesting schemes are implemented in NWP, realizing a coarse spatial grid on global scale to provide BCs for ever smaller grids on continental, country, and regional scales reaching high spatial resolution (Phillips and Shukla, 1973).

In DLWP, nesting has not found application so far and provides a computationally feasible approach for combining global forecasts with local high-resolution dynamics. This would allow DLWP

models to simulate fine scale processes, such as convection leading to the formation of clouds evolving on scales below 5 km (Kendon *et al.*, 2019) and turbulence, even requiring resolutions below 1 km (Bauer *et al.*, 2020).

Physics-Informed DLWP State-of-the-art DLWP models are not guaranteed to generate physically plausible states of the atmosphere due to the lack of physical constraints. As demonstrated in RU III, such constraints have the potential to reduce both the number of parameters and the amount of required training data. At the same time, we showed that reasonably constrained models are robust to out of distribution applications.

These properties are valuable for DLWP, in particular when aiming to generate climate projections on long lead times of decades or longer. An according promising approach of post-processing the outputs of a black-box model is presented in Hansen *et al.* (2023), enforcing the conservation of physical properties, such as mass, energy, or momentum.

Given that global high-resolution NWP models are computationally intractable even for state-of-the-art supercomputers, physics-informed neural networks of the atmosphere form a promising topic for future research; in particular in terms of accelerating the generation of stable and physically plausible forecasts. To date, however, it is still unclear to what extent and on which temporal horizons physical constraints are necessary for weather forecasting; and if they are beneficial after all.

Gradient-Based Inference on Competitive DLWP Exploiting the differentiability of DL models, gradient-based inference techniques on DLWP models have the potential to unveil novel in-

terrelationships between variables and dynamics that are not understood until today, as suggested by Shen *et al.* (2023).

In RU II, we have demonstrated the potential of gradient-based inference, which promises effective knowledge gain in the weather system when applied to competitive DL models, such as our DLWP-HPX developed in RU IV. Seeing that DLWP is about to outperform NWP (Lam *et al.*, 2022; Bi *et al.*, 2023) while depending on less variables, the analysis of according models has large potential for knowledge gain.

Furthermore, investigating latent states of DLWP models on different hierarchies could reveal clusters that relate to particular (global) weather patterns, such as blocking (Nakamura and Huang, 2018; Lupo, 2021) or extreme events (Kautz *et al.*, 2022). Understanding the factors that lead to such clusters relating to atmospheric states and how they transit into other states would be a high-value contribution to atmospheric science.

Miscellaneous DL Applications More promising techniques from the DL science involve *spatial losses*, such as fractional skill score (FSS) (Skok and Roberts, 2016), peak signal-to-noise ratio (PSNR) (Korhonen and You, 2012), structural similarity index measure (SSIM) (Wang *et al.*, 2004), or learning-based measures, such as AtmoDist (Hoffmann and Lessig, 2023), which promise superior over point-wise losses, such as MSE (Lagerquist and Ebert-Uphoff, 2022; Thuemmel *et al.*, 2023).

Since DLWP architectures typically impose multiple hierarchies, they might be trained consecutively, making use of *progressive growth*, which has been demonstrated beneficial for training computer vision models (Fayek *et al.*, 2020).

Furthermore, *second-order optimizers*, such as the limited-memory

Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) (Berahas *et al.*, 2016) and the sharpness-aware minimizer (SAM), have been shown to be more stable and result in models that generalize better due to smoother loss landscapes that prevent overfitting (Chen *et al.*, 2021).

Dedicated *loss weighting* under consideration of variance per latitude, vertical height, and variable type turned out fairly successful in GraphCast and FengWu (Lam *et al.*, 2022; Chen *et al.*, 2023). However, a meaningful explanation of why certain weighting schemes are of value remains pending.

Operating on latent states only instead of performing autoregressive outer-loop rollouts was found helpful in various architectures (DeepMind, 2019; Hafner *et al.*, 2019), since accumulating errors from successive encoding-decoding operations can be prevented. Since all DLWP models are operated autoregressively via outer feedback loops, the simulation of predictions in a *recurrent latent module* constitutes a promising direction for future work, as we have observed substantial improvements in all DISTANA experiments.

Formulating Probabilistic DLWPs Traditionally, forecast uncertainties in NWP are obtained via ensembling methods, which are expensive since a couple of individual deterministic models are evaluated with slightly varying ICs (Palmer, 2019). Competitive DLWP are mostly deterministic and thus employ similar techniques, while DL models for precipitation have been formulated in probabilistic ways already (Espeholt *et al.*, 2021; Gao *et al.*, 2023).

Extending DLWP models with a probabilistic representation of the output would likely resolve the problem of regression to the mean, where DL models generate blurry predictions when trained on long

forecast horizons. Due to the chaotic nature of the atmosphere, deterministic predictions must inevitably approach the mean. Probabilistic models, in contrast, could represent the growing forecast uncertainty with increasing lead time, albeit depending on clever sampling strategies.

Training on Direct Observations A substantial drawback of today's DLWP models lies in their dependency on reanalysis data, such as ERA5 (Hersbach *et al.*, 2020) and from the Coupled Model Intercomparison Project (CMIP) (Meehl *et al.*, 2000). Despite their high quality, reanalysis data do not reflect the real state of the atmosphere but are homogenized by an NWP for spatial and temporal consistency. Accordingly, each DLWP model that is trained on these sources approximates the biases from the numerical models instead of learning to simulate real observations.

A crucial step for DLWP, hence, depends on discarding reanalysis data and training on real observations from remote sensing campaigns (i.e., satellite images), and ground-based measurements. Certainly, this comes with numerous challenges, such as erroneous non-tabular data that are heterogeneously distributed over space. In order to capture the real dynamics of the atmosphere, however, and to allow the extraction and determination of hidden processes that are not washed out by heavy numerical processing, it remains indispensable to train future DLWP models—possibly complemented with reasonable physical constraints—on real-world observations directly.

Appendix A

Publications Contained in this Thesis

A.1 Publication I

A Distributed Neural Network Architecture for Robust Non-Linear Spatio-Temporal Prediction

Matthias Karlbauer[1]*, Sebastian Otte[1],
Hendrik P.A. Lensch[2], Thomas Scholten[3], Volker Wulfmeyer[4], and Martin V. Butz[1]

1- University of Tübingen - Neuro-Cognitive Modeling Group
Sand 14, 72076 Tübingen - Germany

2- University of Tübingen - Computer Graphics
Maria-von-Linden-Straße 6, 72076 Tübingen - Germany

3- University of Tübingen - Soil Science and Geomorphology
Rümelinstraße 19-23, 72070 Tübingen - Germany

4- University of Hohenheim - Institute for Physics and Meteorology
Garbenstraße 30, 70599 Stuttgart - Germany

Abstract.
DISTANA – a distributed spatio-temporal artificial neural network architecture – learns to model and predict spatio-temporal time series dynamics. It learns in a parallel, spatially distributed manner while employing a mesh of recurrent, neural prediction kernels (PKs). Individual PKs predict the local data stream and exchange information laterally. DISTANA essentially assumes that generally applicable causes, which may be locally modified, generate the observed data. We show that DISTANA scales and generalizes to large problem spaces, can approximate complex dynamics, and is robust to overfitting, outperforming other competitive ANNs.

1 Introduction

Modeling and predicting non-linear, spatio-temporal dynamics is challenging for current pattern recognition systems [1]. Representative dynamics include, for example, brain activities [2], video streams [3], traffic flow [4], and weather and climate progressions [5, 6]. The major challenge is to infer, model, and predict the *underlying causes* that generate the perceived data stream. A key property, which all spatio-temporal processes have in common, is that the same underlying causal principles—such as physics when observing natural processes—apply irrespective of time or location. As a result, similar dynamics will be observable repeatedly at different spatial locations and points in time.

DISTANA actively searches for these underlying causes in spatially distributed time series data. It learns a *predictive*, spatio-temporal, neural network *kernel* (PK), which is applied to all nodes of a mesh. Thus, all nodes apply the same operations at different locations. This enables efficient computation in and learning from all nodes in parallel. Moreover, it predisposes DISTANA to

*This work received funding from the German Research Foundation (DFG) under Germany's Excellence Strategy – EXC-Number 2064/1 – Project Number 390727645. Moreover, we thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer.

identify the universal, recurring causes of the observed pattern dynamics. Compared to seven other ANN models, including (temporal) convolutional neural networks (CNNs, TCNs), recurrent neural networks (RNNs), and combinations of both (e.g. ConvLSTM), DISTANA reaches both higher accuracy and robustness at approximating circularly propagating waves. Moreover, it is critically less prone to overfitting and bears the potential to handle heterogeneously distributed sensor meshes. In the near future we will apply DISTANA to related, but more challenging real-world problems, such as modeling the partially chaotic processes that generate our weather and climate.

2 DISTANA

While CNNs can efficiently and accurately process spatially distributed information such as images, RNNs—and long short-term memory cells (LSTMs) [7] in particular—are designed to handle time series data. Recently, Shi et al. [6] proposed ConvLSTM—a convolution-gating architecture, which combines CNNs and LSTMs, thus processing spatial and temporal information simultaneously. GridLSTM [8], on the other hand, extends LSTMs to process not only temporal but also spatial data dimensions sequentially.

DISTANA belongs to a third related class of architectures, which is referred to as graph neural networks (GNNs) [9]. GNNs treat graph vertices and edges in two different neural network components. Unlike earlier GNNs, however, DISTANA integrates LSTM structures, projects the graph, i.e. its mesh, onto a metrical space, and assumes universal causes underlying the observable spatio-temporal data.

DISTANA consists of a PK network, which generates dynamic predictions at each desired spatial location. Multiple PK instances, which share their respective weights, are applied in a sensor mesh, enabling their parallel invocation. Each PK instance receives (1) dynamic input, which is subject to prediction and changes over time, (2) static information, which stays constant and characterizes the location of each PK, and (3) lateral input from neighboring PKs. Typically a PK contains recurrent connections.

3 Experiments

In two experiments, which differ in the data sets used, several ANN architectures including fully connected networks, CNNs, and RNNs are compared with DISTANA. We model a wave-like spatio-temporal process (cf. Figure 1) distributed in a 16×16 mesh. Train and test errors are mean squared errors between network output and target values, which are the dynamic inputs (e.g. wave height) at the next time step. The test error is calculated over 65 time steps of closed loop performance, where the network feeds itself with its own dynamic predictions from the previous time step. The closed loop begins after 15 steps of teacher forcing, which ground the recurrent activity in the network.

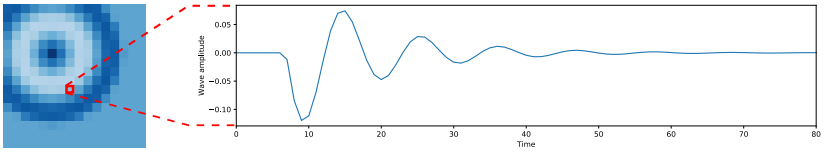


Fig. 1: Data set one. Left: exemplary circular wave. Right: activity pattern over time at one particular position in the two-dimensional wave field.

3.1 Data Set 1

Single sinusoidal waves are generated propagating outwards:

$$u(x, y, t) = \begin{cases} \sin(r_{x,y} - ct) \exp(-d(ct - r_{x,y})) & \text{if } r_{x,y} < ct \\ 0 & \text{else} \end{cases}, \tag{1}$$

where $u(x, y, t)$ is the wave height of the field at a certain position and time, $\sin(r_{x,y} - ct)$ defines the oscillating wave height considering the distance to the wave center $r_{x,y}$ and the current time step t, and $\exp(-d(ct - r_{x,y}))$ causes waves to decay away from the wave origin over time (decay factor $d = 0.25$). Constant $c = 10$ is the wave velocity. Field values that have not been reached by the wave, yet, are set to zero. The waves are not reflected at the borders.

Table 1 shows the performance of all compared models at approximating these circular wave dynamics. Besides *train* and *test errors*, we report the number of *parameters* and the *inference time* of one sequence (consisting of 80 time steps) for each model. In order to rigorously test all models for their generalization abilities, we also trained them on one single sequence (one wave origin) and computed the test error on unseen sequences (test error *1-train-ex.*). Furthermore, to elaborate the models' abilities to approximate variable dynamics, we trained them on waves that travel with varying velocities (test error *var. wave*). *Spatial scalability*, reported in the model descriptions below, indicates whether a model can be directly applied to input fields of different resolutions.

Performances of the following models are compared:

Baselines: *Baseline* $t - 1$ is the identity function; *Baseline zero* predicts zeros.

Fully Connected Networks: A naive and spatially not scalable approach to model the circular wave is a fully connected linear network (*FC-Linear*), with $16 \times 16 = 256$ cells, receiving the flattened input. A more elaborated model is *FC-LSTM*, which replaces the linear layer of *FC-Linear* by a 256-cell LSTM layer to facilitate temporal information processing.

CNN: To reduce the number of parameters, defining a spatially scalable model, numerous *CNN*s with different kernel sizes, a varying number of feature maps, and two convolutional layers were evaluated. The best results, which are reported here, were achieved by using a kernel size of 3×3 and one feature map.

Temporal Convolution Network: TCNs, as a spatially scalable approach, were applied with three 3D convolution layers, each with a $3 \times 3 \times 3$ kernel and $[1, 8, 1]$ feature maps. Other settings did not seem to improve performance.

Table 1: Performance measures of simple wave propagations on data set one.

Model (#pars)	Train error	Test error	Inf. time	1-train-ex.	Var. wave
Baseline $t-1$	-	3.59×10^{-5}	-	3.59×10^{-5}	5.90×10^{-5}
Baseline zero	-	8.88×10^{-5}	-	8.88×10^{-5}	5.84×10^{-4}
FC-Linear (65k)	2.36×10^{-4}	2.56×10^{-4}	**0.0051 s**	2.41×10^{-4}	1.91×10^{-3}
FC-LSTM (524k)	5.34×10^{-5}	1.38×10^{-2}	0.0113 s	2.14×10^{-3}	6.57×10^{-3}
CNN (20)	3.41×10^{-4}	2.22×10^{-4}	0.0115 s	1.37×10^{-3}	2.66×10^{-2}
TCN (2.3k)	1.17×10^{-5}	8.56×10^{-3}	0.0531 s	1.04×10^{-1}	3.09×10^{-2}
CLSTMC (768k)	6.28×10^{-5}	4.67×10^{-1}	0.0230 s	6.13×10^{-4}	2.71×10^{-1}
ConvLSTM1 (144)	1.83×10^{-5}	4.26×10^{-5}	0.0247 s	4.55×10^{-5}	5.85×10^{-4}
ConvLSTM8 (2.9k)	6.34×10^{-6}	$\mathbf{1.28 \times 10^{-6}}$	0.0298 s	1.29×10^{-2}	7.88×10^{-4}
GridLSTM (624)	7.95×10^{-5}	3.62×10^{-1}	5.8786 s	2.86×10^{-1}	1.35×10^{-1}
BiGridLSTM (1.8k)	$\mathbf{6.28 \times 10^{-6}}$	5.65×10^{-1}	11.9900 s	8.67×10^{-1}	4.55×10^{-2}
DISTANA4 (108)	4.18×10^{-5}	2.08×10^{-5}	0.0264 s	1.41×10^{-5}	2.17×10^{-4}
DISTANA26 (2.9k)	2.58×10^{-5}	1.48×10^{-5}	0.0326 s	$\mathbf{2.04 \times 10^{-5}}$	$\mathbf{9.99 \times 10^{-5}}$

CNN-LSTM-CNN (CLSTMC): *CNNs* were extended by inserting a fully connected LSTM layer—making it not spatially scalable—after a variable number of layers. Best results were achieved with one 3×3 convolution followed by a flat LSTM layer and a 3×3 transposed convolution with skip connection.
ConvLSTM: Two models of the spatially scalable ConvLSTM architecture, both with two layers and kernel size three, are reported: *ConvLSTM1* with one feature map in both layers, and *ConvLSTM8* with eight feature maps in the first layer, which are reduced to one in the second layer.
GridLSTM and BiGridLSTM: *GridLSTM* runs forward in time and space; *BiGridLSTM* processes data forward in time but bidirectionally over space. Both are spatially scalable.
DISTANA: DISTANA is spatially scalable. PKs consist of a two-neuron tanh layer, followed by a layer of either four or 26 LSTM cells and another two-neuron tanh layer. This yields, for example, $108 = (2 \cdot 2) + (2 \cdot 4 \cdot 4 + 4 \cdot 4 \cdot 4) + (4 \cdot 2)$ parameters for DISTANA4.

3.2 Data Set 2

To increase data complexity, a second set was created where waves are reflected at borders, such that wave fronts become interactive. We focus our analysis on the most promising architectures determined above. For wave data generation, the two-dimensional wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{2}$$

was solved using the second order central differences approach to obtain an equation for computing the state of the field at a desired position (x, y) in the subsequent time step $t + \Delta_t$

$$u(x, y, t + \Delta_t) = c^2 \Delta_t^2 (u_{xx} + u_{yy}) + 2u(x, y, t) - u(x, y, t - \Delta_t). \tag{3}$$

Table 2: Same evaluation as in Table 1 on data set two, including TCN, ConvLSTM and three variants of DISTANA.

Model (#pars)	Train error	Test error	Inf. time
Baseline $t-1$	-	5.83×10^{-3}	-
Baseline zero	-	1.07×10^{-2}	-
TCN (2.3k)	1.14×10^{-5}	2.11×10^{-1}	0.0707 s
ConvLSTM8 (2.9k)	3.52×10^{-6}	8.09×10^{-2}	0.0289 s
DISTANAv1 (146)	7.89×10^{-6}	8.77×10^{-3}	**0.0280 s**
DISTANAv2 (172)	$\mathbf{1.37 \times 10^{-6}}$	7.68×10^{-4}	0.0294 s
DISTANAv3 (200)	1.64×10^{-6}	$\mathbf{4.99 \times 10^{-4}}$	0.0301 s

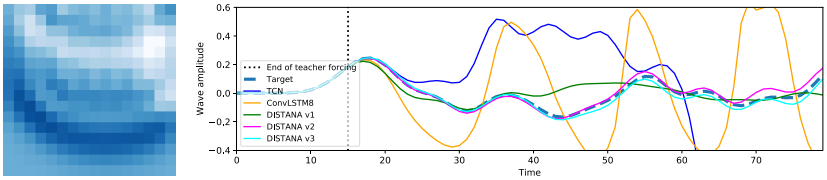


Fig. 2: Data set two. Left: exemplary circular wave with reflecting borders. Right: network dynamics generated by selected architectures.

The unfolding dynamics of higher complexity are much harder to predict (cf. Figure 2). None of the previously tested architectures was able to approximate the dynamics satisfactorily (cf. Table 2) yielding errors larger than the baselines. Accordingly, DISTANA was enhanced as follows:

DISTANA v1: The size of the preprocessing feed forward layer in the PK was increased from two to four neurons.

DISTANA v2: Enhances DISTANA v1 with eight, compared to one, lateral input neurons, which receive input from the eight neighboring PKs, respectively.

DISTANA v3: Enhances DISTANA v2 increasing the number of lateral output neurons from one to eight, dynamically routing individualized outputs to the respective neighbors.

DISTANAv2 and DISTANAv3 strongly outperform the simpler DISTANA versions as well as TCN and ConvLSTM. Table 2 shows that DISTANAv2 reaches the lowest training error, while DISTANAv3 yields the best generalization performance. Fig. 2 shows that when closed loop predictions unfold after 15 steps of teacher forcing, DISTANAv2 and DISTANAv3 approximate the target value still similarly well while the other ANN architectures start to strongly deviate from the target values after only five to ten closed-loop prediction steps. Online video material[1] illustratively shows the further abilities of DISTANA, including its ability to generalize to larger grid sizes.

[1] `https://youtu.be/dH8qcBVuwFg`

4 Discussion

Several ANN architectures were compared at approximating spatio-temporal processes. In the simple scenario, only ConvLSTM and our model, DISTANA, yield smaller test errors than the two baselines when closed loop performance over T prediction time steps is considered. This requires both intrinsic model stability and the maintenance of plausible ongoing dynamics. While the reported test error is in favor of ConvLSTM, DISTANA proved robust to few and variable training data, even with a network that contains only 108 parameters. These findings were corroborated by the evaluations in a second, more complex data set, in which waves were reflected at borders and thus heavily interacted with each other. All other considered architectures failed to generate lasting closed-loop predictions, except for two variants of DISTANA, which consider lateral information propagation explicitly (Figure 2).

Here we have considered regularly distributed grids. However, ongoing work shows that DISTANA can indeed handle irregularly distributed sensor meshes when introducing transition kernels. We thus expect to be able to scale to predict heterogeneously distributed spatiotemporal data, as, for example, necessary to generate highly accurate and further reaching short-range weather forecasts.

References

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[2] Nikola K Kasabov. Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76, 2014.

[3] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[4] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.

[5] J. N. K. Liu, Y. Hu, Y. He, P. W. Chan, and L. Lai. *Information Granularity, Big Data, and Computational Intelligence*, volume 8 of *Studies in Big Data*, chapter Deep Neural Network Modeling for Big Data Weather Forecasting, pages 389–408. Springer, 2015.

[6] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[9] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

A.2 Publication II

Latent State Inference in a Spatiotemporal Generative Model

Matthias Karlbauer[1(✉)], Tobias Menge[1], Sebastian Otte[1], Hendrik P. A. Lensch[2], Thomas Scholten[3], Volker Wulfmeyer[4], and Martin V. Butz[1]

[1] University of Tübingen – Neuro-Cognitive Modeling Group, Sand 14, 72076 Tübingen, Germany
martin.butz@uni-tuebingen.de
[2] University of Tübingen – Computer Graphics, Maria-von-Linden-Straße 6, 72076 Tübingen, Germany
[3] University of Tübingen – Soil Science and Geomorphology, Rümelinstraße 19-23, 72070 Tübingen, Germany
[4] University of Hohenheim – Institute for Physics and Meteorology, Garbenstraße 30, 70599 Stuttgart, Germany

Abstract. Knowledge about the hidden factors that determine particular system dynamics is crucial for both explaining them and pursuing goal-directed interventions. Inferring these factors from time series data without supervision remains an open challenge. Here, we focus on spatiotemporal processes, including wave propagation and weather dynamics, for which we assume that universal causes (e.g. physics) apply throughout space and time. A recently introduced DIstributed SpatioTemporal graph Artificial Neural network Architecture (DISTANA) is used and enhanced to learn such processes, requiring fewer parameters and achieving significantly more accurate predictions compared to temporal convolutional neural networks and other related approaches. We show that DISTANA, when combined with a retrospective latent state inference principle called active tuning, can reliably derive location-respective hidden causal factors. In a current weather prediction benchmark, DISTANA infers our planet's land-sea mask solely by observing temperature dynamics and, meanwhile, uses the self inferred information to improve its own future temperature predictions.

Keywords: Recurrent neural networks · Graph neural networks · Latent inference · Weather prediction

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC number 2064/1 – Project number 390727645. Moreover, we thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer.

I. Farkaš et al. (Eds.): ICANN 2021, LNCS 12894, pp. 384–395, 2021.
https://doi.org/10.1007/978-3-030-86380-7_31

1 Introduction

When considering our planet's weather, centuries of past research have identified a large number of factors that affect its highly nonlinear and partially chaotic dynamics. Yet, can we ever be sure of having identified all hidden causal factors? Moreover, do we have (sufficient) data about them? These are fundamental questions in any prediction or forecasting task, including spatiotemporal processes such as soil property dynamics, traffic forecasting, energy-flow prediction (e.g. in brains or supply networks), or recommender systems. Here, we investigate how unobservable hidden factors may be inferred from spatiotemporal data streams.

When regularities in hidden causes are detectable, they may be encoded in the latent activities of recurrent neural networks [18,20], such as a long short-term memory (LSTM) [10]. The involved and conventional forward-directed inference of recurrent neural networks, however, has two main disadvantages: First, the encodings of the hidden causes form while streaming data, meaning they are not available from the beginning of a sequence. Second, learning, detecting and shaping the encodings is relatively hard, because the error signal only decreases once the unfolding data stream is suitably compressed.

To overcome these limitations, we combine and extend the recently introduced DIstributed SpatioTemporal graph Artificial Neural network Architecture (DISTANA) [13] with active tuning (AT) [7,8,17], which facilitates the determination of hidden causal states via retrospective inference over time. Projected onto stable neural states, akin to parametric bias neurons [23,24], AT searches for constant input biases, assuming that the observed dynamics are influenced by particular constant and only indirectly observable factors.

Following the idea of *relational inductive biases* [3], DISTANA is designed to model the hidden causal processes that generate spatiotemporal dynamics. Hence, DISTANA assumes that the sensed dynamics are generated by universal causal principles (e.g. physics). Moreover, we endow DISTANA with the expectation that constant, hidden factors modify the spatiotemporal processes locally. For example, weather dynamics follow the universal principles of thermodynamics from physics and are locally dependent on the topology.

The contributions we make are as follows: (A) the combination of DISTANA with active tuning (AT) to infer constant, hidden factors locally, even when these factors are never made available to the network – neither as input nor as (target) output. (B) we show that reasonable latent neural activities are inferred during training and testing via retrospective spatiotemporal analysis. (C) after having learned a distributed, generative model of the globally unfolding dynamics, we demonstrate that our planet's land-sea mask as well as other causal factors can be inferred via the retrospective analysis of unfolding weather dynamics – partially again even when the algorithm was never informed about these factors – to increase the model's prediction abilities.

We conclude that the retrospective inference of latent states via AT offers a promising method to identify hidden factors in data streams, and that graph neural networks (GNN) like DISTANA bear great potential at modeling real-world spatiotemporal processes.

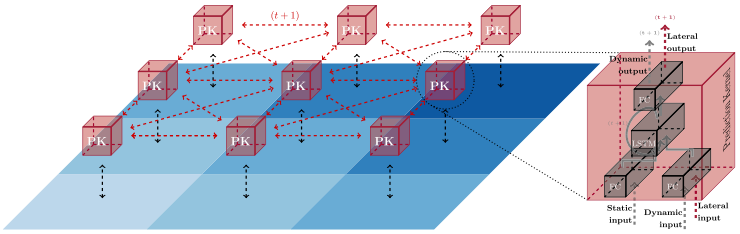


Fig. 1. 3×3 sensor mesh grid showing the connection scheme of Prediction Kernels (PKs) that model the local dynamical process while communicating laterally. Figure modified from [14].

2 DISTANA

As introduced in [13] and following the naming convention of [27], DISTANA (the DIstributed SpatioTemporal graph Artificial Neural network Architecture) can be described as a spatiotemporal graph neural network (ST-GNN). While GNNs give the designer a large amount of freedom in controlling the flow of information within the model (referred to as relational inductive biases) [3], they are reported to model physical systems with very high precision and accuracy for up to several hundreds of time steps even during closed loop prediction [2, 16,21,22,25]. Thorough surveys about GNNs and the numerous ways of creating the graph and setting up the connection schemes are written by [3,6,27].

The GNN used in this work, DISTANA, consists of prediction kernels (PKs), which are arranged in a lattice structure. PKs model local dynamics concurrently. In every time step t, each PK receives (i) local dynamic data, and (ii) lateral output activities from the neighboring PKs from $t-1$ to exchange information between PKs. Here, we extend the PKs to (iii) additionally receive location specific static inputs. The time recurrent PKs process this information, combine it with their previous latent state, and generate (i) predictions of the next local dynamic data input at $t+1$, as well as (ii) outputs to the laterally connected PKs (cf. Fig. 1). PKs are akin to a spatiotemporal convolutional kernel, since all PKs share identical weights, that is, a single set of weights is applied and optimized in every grid cell. As a result, the likelihood of overfitting local data irregularities is reduced and the emergence of a highly generalizing and universally applicable set of weights is fostered. Because of the reduction of trainable weights, less data is needed for training.

2.1 Alternative State-of-the-Art Architectures

We compared DISTANA with two well-suited deep learning approaches. First, we tested convolutional long short-term memory models (ConvLSTMs) [28] to predict circular wave dynamics (see Fig. 2). The used ConvLSTM model has

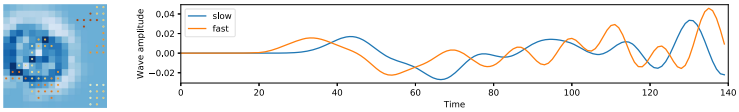


Fig. 2. Left: two-dimensional wave propagating through a 16×16 grid with obstacles. Darker dots in the grid nodes correspond to stronger blocking effect on the wave. Right: wave activity for two exemplary positions in the grid, with fast and slow propagation speeds.

2 952 free parameters to project the $16 \times 16 \times 1$ input (ignoring batch and time dimensions) via the first layer on eight feature maps (resulting in dimensionality $16 \times 16 \times 8$) and subsequently via the second layer back to one output feature map. All kernels have a filter size of $k = 3$, apply zero-padding and are implemented with a stride of one. The code was taken and adapted from[1]. Second, we tested Temporal Convolution Networks (TCN) [1,9,12]. The TCN used in this work is a three-layer network with 2 306 parameters, where the input layer projects to eight feature maps, which project their values back to one output value. A kernel filter size of $k = 3$ is used for the two spatial dimensions in combination with the standard dilation rate of $d = 1, 2, 4$ for the temporal dimension, resulting in a temporal horizon of 28 time steps, cf. [1]. Various experiments with other sizes and deeper network structures have not yielded any better performance than the one reported. Code was taken and adapted from [1].

2.2 Static Input Inference via Active Tuning

Essentially, active tuning (AT) [7,8,17] can be seen as a different paradigm for handling RNNs: instead of the usual input → compute → output scheme, a subset of the RNN's neurons is decoupled from the direct input signal. The activation of this subset of neurons is computed from the RNN's prediction-based gradients, both during training and testing. Gradient information is obtained from backpropagating the discrepancy between the RNN's predicted output $\hat{\mathbf{y}}$ and the desired output $\mathbf{y}$. Thus, the neuron dynamics of the subset is solely influenced by the target indirectly, by means of temporal gradient information induced by the prediction error.

In this work, as mentioned before, DISTANA receives dynamic and lateral input, while the static input $\mathbf{s}$ is withheld and must be inferred via AT to reasonably model the unfolding dynamics. Technically, $\mathbf{s}$ is fed to the model initially as a zero vector and optimized iteratively through the AT method. AT is applied to reduce local prediction errors, while the PK weights are updated as usual to reduce global prediction errors and model universal dynamics.

The active tuning algorithm, please refer to [17] for more information, can be applied in combination with any desired gradient optimization strategy, e.g. Adam [15]. Furthermore, an arbitrary number of optimization cycles c, here

[1] https://github.com/ndrplz/ConvLSTM_pytorch.

$c = 1$, and history length H, here $H = 10$, can be chosen, where the latter indicates up to what time in the local past the latent context vector $\mathbf{s}$, which is assumed to be constant, is optimized. The AT optimization procedure is realized every ten time steps retrospectively on the predicted dynamic input, starting from time step τ to find a converging $\mathbf{s}$.

We have modified (AT) for application on two-dimensional data in order to infer an individual, slowly changing local latent variable, denoted as $\mathbf{s}_i$, for each vertex of the two-dimensional grid. AT so far has been applied to one-dimensional time series prediction for the inference of rarely changing contextual [7,8] or dynamically changing latent states [17]. In contrast to previous applications of AT, the inferred local static input $\mathbf{s}$ is not reset between sequences during training here, assuming a constant static context.

In our initial experiments, the inferred static input frequently drifted or potentially exploded, comparable to an Intern Covariate Drift [11]. We solve this problem, similarly to [11], by normalizing the inferred latent variable (in our case the static input $\mathbf{s}$), via the mean μ_s and standard deviation σ_s with respect to all inferred static inputs $\{s_i^t\}_{i=1}^k$, where k is the number of cells or pixels in the two-dimensional field. Additionally, to remove noise from the inference process caused by inconsistent gradient signals before and after the normalization, the weights W and the bias neuron b of the static input preprocessing layer are modified such that the activation of the static input preprocessing layer remains the same before and after the normalization:

$$b \leftarrow b + W \cdot \mu_s; \qquad W \leftarrow W \cdot \sigma_s \tag{1}$$

While the activation of the network is preserved, the gradients backpropagated through the static input preprocessing layer are affected asymmetrically by the modified weights and bias. In our experiments, this has been shown to substantially improve both the inference during training and the convergence of the inference process during testing.

3 Experiments and Results

The experiments are based on two classes of spatiotemporal time series. Both are representatives of universal, but locally and temporally modifiable, spatiotemporal, causal processes that propagate dynamics over local topologies throughout a homogeneously connected graph.

3.1 2D Circular Wave

Following [13], a spatiotemporal wave propagation dataset was created to validate our approach. In comparison to [13], however, the data generation was enhanced such that the wave propagation velocity could be contextually modified locally, which intuitively resembles obstacles in the water, which affect the wave's propagation behavior (cf. Fig. 2).

This benchmark was used to (a) demonstrate and compare DISTANA's principal capability to model locally parameterized spatiotemporal dynamics and (b) determine whether DISTANA can be used in combination with AT to infer an underlying and hidden static (causal) factor, which modifies the observed dynamics locally. Adam [15] is used for training with a learning rate of 10^{-3} along with Scheduled Sampling [4] with a linear slope of 270 epochs, transitioning from a probability of $0.0 \rightarrow 0.9$ of feeding the network with its own output in the next iteration instead of the teacher signal. During each sequence, 30 teacher forcing steps are conducted to induce reasonable network activities before switching to closed loop. Network inputs $\mathbf{x}$ and the according targets $\mathbf{y}$ are exactly the same sequences shifted by one time step to train four different model types (ConvLSTM, TCN, DISTANA and DISTANA + AT) to iteratively predict the next two-dimensional dynamic wave field state (one step ahead prediction). For the static input inference, DISTANA is augmented with a parametric bias neuron, whose activity is inferred during training and testing, aiming at the identification of an unknown location-specific wave velocity-influencing factor (static context). Training was realized over 300 epochs consisting of 100 training sequences of length 120 each. The target static context vector $\mathbf{s} \in \mathbb{R}^{16\times16}$ was initialized by drawing values from $\{0.2, 0.3, 0.5, 0.6, 0.8, 0.9\}$, where small values cause the waves to propagate slower at the according pixel. An exemplary ground truth context map $\mathbf{s}_{GT}$ is visualized in Fig. 2 (left, brownish dots). Note that $\mathbf{s}_{GT}$ was used for the data generation but has never been provided to any model. The preprocessing layer size of DISTANA was set to eight neurons and the subsequent LSTM layer consisted of twelve cells, yielding 1 236 parameters. For the DISTANA + AT model, an additional static preprocessing layer with five neurons was used, resulting in 1 486 weights overall, compared to 2 952 and 2 306 weights for ConvLSTM and TCN, respectively.

To test the models' generalization capabilities, 16 new static context vectors $\mathbf{s}'$ have been generated by drawing from $\{0.2, 0.3, \ldots, 1.0\}$ (e.g. see Fig. 3, top right-most). All models were evaluated on 50 sequences – made up of 120 time steps each – per $\mathbf{s}'$. Reasonable activity was induced into the models by applying 30 steps of teacher forcing, followed by 90 steps of closed loop prediction for which an average MSE over all test examples and spatial locations was computed. For DISTANA + AT, the static context has been inferred before the testing on 50 separate sequences, using a history length of $H = 30$, one optimization cycle ($c = 1$), and an inference learning rate of $\eta = 0.1$ for the first three epochs, and $\eta = 0.01$ for the remaining seven epochs.

3.2 2D Circular Wave Results

The prediction accuracy of ConvLSTM, TCN, DISTANA and DISTANA + AT differs considerably. TCN without scheduled sampling tends to start oscillating increasingly after few steps of closed loop prediction, resulting in a mediocre MSE score of $(2.94 \pm 238) \times 10^{-2}$, while ConvLSTM $(8.69 \pm 0.87) \times 10^{-4}$ and DISTANA $(8.69 \pm 0.87) \times 10^{-4}$, both trained with scheduled sampling, tend to vanish after few steps of closed loop prediction. Solely DISTANA + AT trained

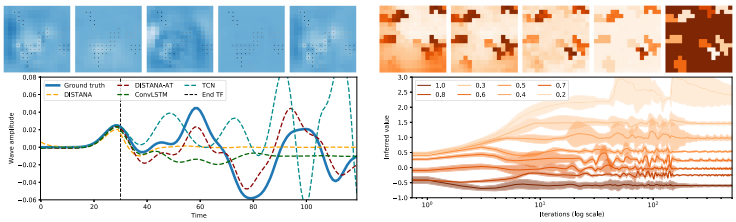


Fig. 3. Top left: ground truth and model outputs at time step 80, which is 30 time steps after the start of closed loop prediction (from left to right: ground truth, ConvLSTM, TCN, DISTANA, DISTANA + AT). Bottom left: ground truth and model outputs over time at position $x = 2$, $y = 6$ in the two-dimensional grid. Top right: inferred static context during testing with values in the range $[-0.6, 2.6]$ after 1, 2, 10, 500 iterations and ground truth with values in $[0.0, 1.0]$. For the ground truth, darker color corresponds to a stronger blocking effect on the wave, which was learned and inferred inversely by the network. Bottom right: average inferred contexts over time during testing (x-axis log scaled).

with scheduled sampling is able to preserver a stable activation pattern with an MSE of $(3.87 \pm 2.48) \times 10^{-4}$.

Furthermore, as shown in Fig. 3 (bottom right), DISTANA + AT preserves a linear ordering when inferring context values that were never encountered during training as indicated by the static context values 0.4 and 0.7, which are properly mapped to roughly -0.1 and -1.1, respectively, without violating the propagation speed order with respect to other static context values. Thus, looking at the estimated static context $\hat{\mathbf{s}}$, it turns out that the latent state inferred by AT correctly reproduced the monotonicity of the here known underlying structure. The static context map at test time, which is different to the map on which the model was trained on, is inferred correctly (see image sequence of Fig. 3, top right). When comparing the prediction accuracy of DISTANA and DISTANA + AT in Fig. 3 (top and bottom left), the self-inferred static context clearly helps DISTANA + AT to model the two-dimensional wave.

3.3 WeatherBench

Recently, [19] introduced a benchmark for comparing mid-range (that is three to five days) weather forecast qualities of data driven and physics-based approaches. While globally regularly aggregated data are provided in three spatial resolutions (5.625°, 2.8125° and 1.40525° resulting in 32×64, 64×128 and 128×256 grid points, respectively), evaluated baselines are reported for the coarsest resolution only, which in consequence we chose too for elaborating and comparing DISTANA. Baselines are generated by means of persistence (tomorrow's weather is today's weather), climatology, linear regression, and physics-based numerical weather prediction models. Moreover, convolutional neural networks (CNNs) are

either applied iteratively or directly. Baselines are computed solely on three or five day predictions of the geopotential at an atmospheric pressure level of 500 hPa (roughly at 5.5 km height, called Z500) and the temperature at 850 hPa (∼1.5 km height, referred to as T850). Beyond Z500 and T850, weatherBench consists of numerous additional dynamic variables (humidity, precipitation, wind direction and speed, solar radiation, etc.), partially reported on multiple vertical layers, and static variables (land-sea mask, soil type, orography, latitude and longitude).

We use weatherBench (a) to explore DISTANA's abilities to approximate real-world phenomena by comparing it to [19]'s iterative CNN approach and (b) to investigate how to apply gradient-based inference techniques in order to infer local static context (e.g. the land-sea mask) that affect Z500 and T850. The experiments we conducted on weatherBench focused on the prediction of the Z500 (geopotential) and T850 (temperature) variables. DISTANA and DISTANA + AT were trained for 2 000 epochs on weather data from 1979, using a learning rate of 10^{-4}, validated on 2016, and tested on 2017. Each year was partitioned into sequences of 96 hourly steps, yielding 91 sequences per year. Increasing the set sizes or changing the training, validation, or testing years did not seem to alter the results or model performances. DISTANA's preprocessing and LSTM layers were set to 50 neurons and cells, respectively. Furthermore, the implementation of DISTANA was enhanced to support a varying lateral communication vector size, which then was increased from one to five neurons, to enable neighboring PKs to exchange information of higher complexity, yielding ∼25 000 parameters, slightly varying with the number of input variables. Moreover, the lateral connection scheme of DISTANA was specified such that information exiting the horizontal boundaries would enter at the other end of the field to match weatherBench's horizontally connected spherical data composition.

Selected static information provided by weatherBench was adapted and extended to facilitate the learning process. Changes were made to the latitude and longitude variables: latitude was transformed to be zero at the equator and non-linearly rising to one towards the poles, based on cos(lat). The longitude variable was split into its sine and cosine component, creating a circular encoding to match the spherical shape of the Earth from which the data origins. Additionally, one-dimensional north- and south-flags were provided to account for the missing neighbors in the north- and south-most rows in the grid. As has been done in [26], we also provide the top of atmosphere total incident solar radiation (tisr). All variables were normalized to the range of $[-1.0, 1.0]$. When using AT to infer a latent static context $\tilde{\mathbf{s}}$, the values were clamped to $[-1.0, 1.0]$ to prevent them from drifting or exploding. If not specified differently, we provide the models with the dynamic variable Z500 or T850 (being subject for prediction), along with nine static inputs: orography, land-sea mask (LSM), soil type, longitude (two-dimensional), latitude, tisr, and the north- and south-end flags.

3.4 WeatherBench Results

The evaluation of DISTANA being trained to predict the Z500 variable for a lead time of 72 h yielded an RSME of 816, which is better than the current best

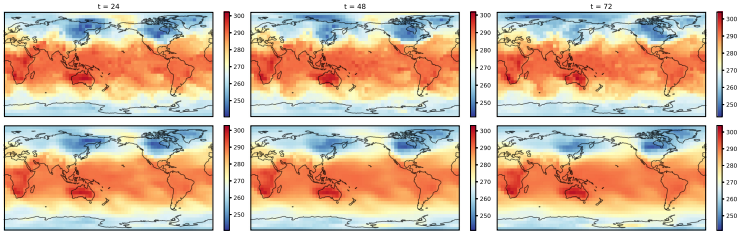


Fig. 4. Predicted temperature (T850) in degree Kelvin for 24, 48 and 72 h (corresponding to time steps) into the future (closed-loop). The first row shows the ground truth and the second row the network output.

comparable iterative approach reported on the benchmark (RMSE = 1114). However, seeing that the best numerical operational weather prediction model produces an RMSE of 154 and other machine learning approaches achieve an RMSE of 268, there is certainly room for improvement. Nonetheless, DISTANA offers the best learned generative, iterative processing model on the benchmark without applying techniques that reduce the distortion resulting from transforming the spherical Earth data to a regular two-dimensional grid.

A second experiment was conducted to (a) investigate whether DISTANA + AT is able to predict the T850 variable, see Fig. 4, and (b) simultaneously infer missing land-sea mask (LSM) values only from the observed T850 dynamics. The model thus received the same static input as in the previous experiment along with the T850 variable during training. However, only two thirds of the LSM values were provided. The other third, considered missing values, which covered America and the Atlantic ocean, were to be inferred. After training, the entire LSM vector $\hat{\mathbf{s}}_{\text{LSM}}$, initialized with zeros, was retrospectively tuned via AT such that it would best explain the observed dynamics. As visualized in Fig. 5 (top center) the missing LSM is inferred reasonably, including the American continent, which the network has never seen during training or inference. These findings suggest that the model learned a generalizable, globally applicable encoding of the LSM's influence on the T850 dynamics.

In a third experiment, we used an additional latent neuron – a parametric bias neuron – that is locally tuned during training via AT. This latent neuron is supposed to be tuned freely by the model to develop any code that helps the model to predict the observed dynamics. We were particularly interested in evaluating whether DISTANA would develop latent states $\tilde{\mathbf{s}}$ that distinctively encode prediction-relevant, hidden causal factors that correspond to observable values. For example, we wanted to see whether DISTANA would develop a latent code that resembles any land-coding quantity. Thus, in this experiment, we try to answer the question what latent states are inferred depending on the predicted variable and how the presence of land-relevant input does affect the generation of this latent code.

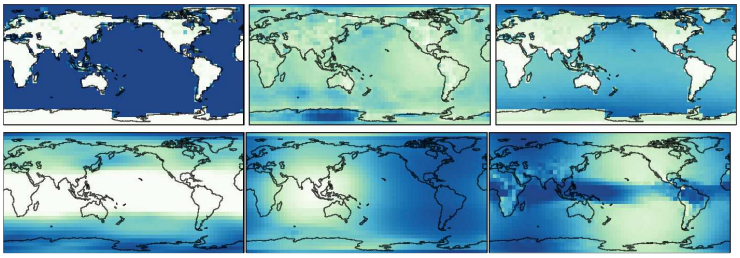


Fig. 5. Top left: original land-sea mask (LSM). Top center: global LSM inferred during testing after being trained on two thirds of the globe (the model has never seen America's LSM). Top right: a latent vector which developed during training and encodes LSM information as well as a decent latitude coding. Bottom: three latent variable codes that freely emerged during training of the Z500 (left, center) and T850 (right) variables.

Our results indicate that the nature of the developed latent states depends considerably on both the variable that is subject for prediction (Z500 or T850) and the additional static data provided. Figure 5 (top right) shows a clear tendency to encode land-sea information, augmenting it with a latitude code, when all previously mentioned static inputs (including LSM) were provided. When training a model to predict Z500, the emerging latent variables rather seem to encode latitude, albedo, monsoon [5], or humidity-distribution patterns (Fig. 5 bottom left and center). Excitingly, nuances of LSM and orography become visible when training to predict T850 without receiving any land-coding inputs (see Fig. 5 bottom right). Nevertheless, further studies are necessary to verify to which extent the inferred variables correlate with observations in detail.

4 Final Discussion

The presented results indicate that the combination of the DIstributed, SpatioTemporal graph Artificial Neural network Architecture, DISTANA, with the retrospective inference mechanism called active tuning (AT), bears large potential at predicting spatiotemporal real-world phenomena (e.g. weather). It outperforms competing deep learning algorithms by generating more accurate closed-loop predictions into the future. In addition, it can infer hidden causes by mere observation of a dynamic process. In particular, AT in DISTANA is well-suited for inferring (i) contrastive hidden causes during learning and (ii) hidden static activities while minimizing loss online. While we believe that these hidden factors tend to identify causal influences – because they form for improving the accuracy of the predicted dynamics – future research will need to investigate the robustness of this tendency.

During learning, cumulative error signals in latent parametric bias neurons at the individual prediction kernels tend to develop encodings of hidden, dynamic-influencing factors. To a certain extent, these neuronal encodings resemble physical properties, such as albedo or the land-sea mask, depending on the type of dynamics that is to be predicted (e.g. temperature or geopotential). That is, the projection of the gradient onto static neural activities identifies local parametric bias activities that best characterize local, hidden causal factors.

Overall, the results suggest that our approach of assuming and inferring hidden causes with constrained properties – such as being locally distinct, constant, but universally present – offers strong potential in fostering the development of process-explaining structures.

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
2. Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al.: Interaction networks for learning about objects, relations and physics. In: Advances in Neural Information Processing Systems, pp. 4502–4510 (2016)
3. Battaglia, P.W., et al.: Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261 (2018)
4. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. arXiv:1506.03099 (2015)
5. Boers, N., Goswami, B., Rheinwalt, A., Bookhagen, B., Hoskins, B., Kurths, J.: Complex networks reveal global pattern of extreme-rainfall teleconnections. Nature **566**(7744), 373–377 (2019)
6. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
7. Butz, M.V., Bilkey, D., Humaidan, D., Knott, A., Otte, S.: Learning, planning, and control in a monolithic neural event inference architecture. Neural Netw. **117**, 135–144 (2019)
8. Butz, M.V., Menge, T., Humaidan, D., Otte, S.: Inferring event-predictive goal-directed object manipulations in REPRISE. In: Tetko, I.V., Kurková, V., Karpov, P., Theis, F. (eds.) ICANN 2019. LNCS, vol. 11727, pp. 639–653. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30487-4_49
9. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 933–941. JMLR. org (2017)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, Lille, France, 07–09 July 2015, vol. 37, pp. 448–456. PMLR (2015)
12. Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A.V.D., Graves, A., Kavukcuoglu, K.: Neural machine translation in linear time. arXiv:1610.10099 (2016)

13. Karlbauer, M., Otte, S., Lensch, H.P.A., Scholten, T., Wulfmeyer, V., Butz, M.V.: A distributed neural network architecture for robust non-linear spatio-temporal prediction. arXiv:1912.11141 (2019)
14. Karlbauer, M., Otte, S., Lensch, H.P.A., Scholten, T., Wulfmeyer, V., Butz, M.V.: Inferring, predicting, and denoising causal wave dynamics. In: Farkaš, I., Masulli, P., Wermter, S. (eds.) ICANN 2020. LNCS, vol. 12396, pp. 566–577. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61609-0_45
15. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations, December 2014
16. Kipf, T., Fetaya, E., Wang, K.C., Welling, M., Zemel, R.: Neural relational inference for interacting systems. arXiv:1802.04687 (2018)
17. Otte, S., Karlbauer, M., Butz, M.V.: Active tuning. arXiv:2010.03958 (2020)
18. Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S.M.A., Botvinick, M.: Machine theory of mind. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, Stockholm, Sweden, 10–15 July 2018, vol. 80, pp. 4218–4227. PMLR (2018)
19. Rasp, S., Dueben, P.D., Scher, S., Weyn, J.A., Mouatadid, S., Thuerey, N.: WeatherBench: a benchmark dataset for data-driven weather forecasting. arXiv:2002.00469 (2020)
20. Rodriguez, R.C., Alaniz, S., Akata, Z.: Modeling conceptual understanding in image reference games. In: Advances in Neural Information Processing Systems, pp. 13155–13165 (2019)
21. Sanchez-Gonzalez, A., et al.: Graph networks as learnable physics engines for inference and control. arXiv:1806.01242 (2018)
22. Santoro, A., et al.: A simple neural network module for relational reasoning. In: Advances in Neural Information Processing Systems, pp. 4967–4976 (2017)
23. Sugita, Y., Tani, J., Butz, M.V.: Simultaneously emerging Braitenberg codes and compositionality. Adapt. Behav. **19**, 295–316 (2011)
24. Tani, J., Ito, M., Sugita, Y.: Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB. Neural Netw. **17**, 1273–1289 (2004)
25. Van Steenkiste, S., Chang, M., Greff, K., Schmidhuber, J.: Relational neural expectation maximization: unsupervised discovery of objects and their interactions. arXiv:1802.10353 (2018)
26. Weyn, J.A., Durran, D.R., Caruana, R.: Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. arXiv:2003.11927 (2020)
27. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv:1901.00596 (2019)
28. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems, pp. 802–810 (2015)

A.3 Publication III

Composing Partial Differential Equations with Physics-Aware Neural Networks

Matthias Karlbauer[⋆ 1] **Timothy Praditia**[⋆ 2] **Sebastian Otte**[1] **Sergey Oladyshkin**[2] **Wolfgang Nowak**[2] **Martin V. Butz**[1]

Abstract

We introduce a compositional physics-aware FInite volume Neural Network (FINN) for learning spatiotemporal advection-diffusion processes. FINN implements a new way of combining the learning abilities of artificial neural networks with physical and structural knowledge from numerical simulation by modeling the constituents of partial differential equations (PDEs) in a compositional manner. Results on both one- and two-dimensional PDEs (Burgers', diffusion-sorption, diffusion-reaction, Allen–Cahn) demonstrate FINN's superior modeling accuracy and excellent out-of-distribution generalization ability beyond initial and boundary conditions. With only one tenth of the number of parameters on average, FINN outperforms pure machine learning and other state-of-the-art physics-aware models in all cases—often even by multiple orders of magnitude. Moreover, FINN outperforms a calibrated physical model when approximating sparse real-world data in a diffusion-sorption scenario, confirming its generalization abilities and showing explanatory potential by revealing the unknown retardation factor of the observed process.

1. Introduction

Artificial neural networks (ANNs) are considered universal function approximators (Cybenko, 1989). Their effective learning ability, however, greatly depends on domain and task-specific prestructuring and methodological modifications referred to as inductive biases (Battaglia et al., 2018). Typically, inductive biases limit the space of possible models by reducing the opportunities for computational shortcuts that might otherwise lead to erroneous implications derived from a potentially limited dataset (overfitting). The recently evolving field of physics-informed machine learning employs physical knowledge as inductive bias providing vast generalization advantages in contrast to pure machine learning (ML) in physical domains (Raissi et al., 2019). While numerous approaches have been introduced to augment ANNs with physical knowledge, these methods either do not allow the incorporation of explicitly defined physical equations (Long et al., 2018; Seo et al., 2019; Guen & Thome, 2020; Li et al., 2020a; Sitzmann et al., 2020) or cannot generalize to other initial and boundary conditions than those encountered during training (Raissi et al., 2019).

In this work, we present the FInite volume Neural Network (FINN)—a physics-aware ANN adhering to the idea of spatial and temporal discretization in numerical simulation. FINN consists of multiple neural network modules that interact in a distributed, compositional manner (Lake et al., 2017; Battaglia et al., 2018; Lake, 2019). The modules are designed to account for specific parts of advection-diffusion equations, a class of partial differential equations (PDEs). This modularization allows us to combine two advantages that are not yet met by state-of-the-art models: The explicit incorporation of physical knowledge and the generalization over initial and boundary conditions. To the best of our knowledge, FINN's ability to adjust to different initial and boundary conditions and to explicitly learn constitutive relationships and reaction terms is unique, yielding excellent out-of-distribution generalization. The core contributions of this work are:

- Introduction of FINN, a physics-aware ANN, explicitly designed to generalize over initial and boundary conditions, yielding excellent generalization ability.
- Evaluation of state-of-the-art pure ML and physics-aware models, contrasted to FINN on one- and two-dimensional benchmarks, demonstrating the benefit of explicit model design.
- Application of FINN to a real-world contamination-diffusion problem, verifying its applicability to real, spatially and temporally constrained training data.

[⋆]Equal contribution [1]Neuro-Cognitive Modeling, University of Tübingen, Tübingen, Germany [2]Department of Stochastic Simulation and Safety Research for Hydrosystems, University of Stuttgart, Stuttgart, Germany. Correspondence to: Matthias Karlbauer <matthias.karlbauer@uni-tuebingen.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2. Related Work

Non-Physics-Aware ANNs Pure ML models that are designed for spatiotemporal data processing can be separated into temporal convolution (TCN, Kalchbrenner et al., 2016) and recurrent neural networks. While the former perform convolutions over space and time, representatives of the latter, e.g., convolutional LSTM (ConvLSTM, Shi et al., 2015) or DISTANA (Karlbauer et al., 2019), aggregate spatial neighbor information to further process the temporal component with recurrent units. Since pure ML models do not adhere to physical principles, they require large amounts of training data and parameters in order to approximate a desired physical process; but still are not guaranteed to behave consistently outside the regime of the training data.

Physics-Aware ANNs When designed to satisfy physical equations, physics-aware ANNs are reported to have greater robustness in terms of physical plausibility. For example, the physics-informed neural network (PINN, Raissi et al., 2019) consists of an MLP that satisfies an explicitly defined PDE with specific initial and boundary conditions, using automatic differentiation. However, the remarkable results beyond the time steps encountered during training are limited to the very particular PDE and its conditions. A trained PINN cannot be applied to different initial conditions, which limits its applicability in real-world scenarios.

Other methods by Long et al. (PDENet, 2018), Guen & Thome (PhyDNet, 2020), or Sitzmann et al. (SIREN, 2020) learn the first n derivatives to achieve a physically plausible behavior, but lack the option to include physical equations. The same limitation holds when operators are learned to approximate PDEs (Li et al., 2020a;b), or when physics-aware graph neural networks are applied (Seo et al., 2019). Yin et al. (APHYNITY, 2020) suggest to approximate equations with an appropriate physical model and to augment the result by an ANN, preventing the ANN to approximate a distinct part within the physical model. For more comparison to related work, please refer to subsection B.1.

In summary, none of the above methods can explicitly learn particular constitutive relationships or reaction terms while simultaneously generalizing beyond different initial and boundary conditions.

3. Finite Volume Neural Network (FINN)

Problem Formulation Here, we focus on modeling spatiotemporal physical processes. Specifically, we consider systems governed by advection-diffusion type equations (Smolarkiewicz, 1983), defined as:

$$\frac{\partial u}{\partial t} = D(u)\frac{\partial^2 u}{\partial x^2} - v(u)\frac{\partial u}{\partial x} + q(u), \tag{1}$$

where u is the quantity of interest (e.g. salt distributed in water), t is time, x is the spatial coordinate, D is the diffusion coefficient, v is the advection velocity, and q is the source/sink term. Eq. 1 can be partitioned into three parts: The storage term $\frac{\partial u}{\partial t}$ describes the change of the quantity u over time. The flux terms are the advective flux $v(u)\frac{\partial u}{\partial x}$ and the diffusive flux $D(u)\frac{\partial^2 u}{\partial x^2}$. Both calculate the amount of u exchanged between neighboring volumes. The source/sink term $q(u)$ describes the generation or elimination of the quantity u. Eq. 1 is a general form of PDEs with up to second order spatial derivatives, but it has a wide range of applicability due to the flexibility of defining $D(u)$, $v(u)$, and $q(u)$ as different functions of u, as is shown by the numerical experiments in this work.

The finite volume method (FVM, Moukalled et al., 2016) discretizes a simulation domain into control volumes ($i = 1, \dots, N_x$), where exchange fluxes are calculated using a surface integral (Riley et al., 2009). In order to match this structure in FINN, we introduce two different kernels, which are (spatially) casted across the discretized control volumes: the flux kernel, modeling the flux terms (i.e. lateral quantity exchange), and the state kernel, modeling the source/sink term as well as the storage term. The overall FINN architecture is shown in Figure 1.

Flux Kernel The flux kernel $\mathcal{F}$ approximates the surface integral for each control volume i with boundary Ω by a composition of multiple subkernels f_j, each representing the flux through a discretized surface element j:

$$\mathcal{F}_i = \sum_{j=1}^{N_{s_i}} f_j \approx \oint_{\omega \subseteq \Omega} \left(D(u)\frac{\partial^2 u}{\partial x^2} - v(u)\frac{\partial u}{\partial x} \right) \cdot \hat{n}\, d\Gamma, \tag{2}$$

where N_{s_i} is the number of discrete surface elements of control volume i, ω is a continuous surface element (a subset of Ω), f_j are subkernels (consisting of feedforward network modules), and $\hat{n}$ is the unit normal vector pointing outwards of ω.

In our exemplary one-dimensional arrangement, two subkernels f_{i-} and f_{i+} (see Figure 1) contain the modules $\varphi_{\mathcal{D}}$, $\varphi_{\mathcal{A}}$, and $\varphi_{\mathcal{N}}$. Module $\varphi_{\mathcal{N}}$ is a linear layer with the purpose to approximate the first spatial derivative $\frac{\partial u}{\partial x}$, i.e.

$$\frac{\partial u_i}{\partial x} \approx \begin{cases} \varphi_{\mathcal{N}}(u_i, u_{i-1}) & \text{on } f_{i-} \\ \varphi_{\mathcal{N}}(u_i, u_{i+1}) & \text{on } f_{i+} \end{cases}. \tag{3}$$

Technically, $\varphi_{\mathcal{N}}$ is supposed to learn the numerical FVM stencil, being nothing else but the difference between its inputs, i.e. the quantity at two neighboring control volumes in ideal one-dimensional problems. This signifies that the weights of $\varphi_{\mathcal{N}}$ should amount to $[-1, 1]$ with respect to $[u_i, u_{i-1}]$ and $[u_i, u_{i+1}]$ in order to output their difference.

Composing Partial Differential Equations with Physics-Aware Neural Networks

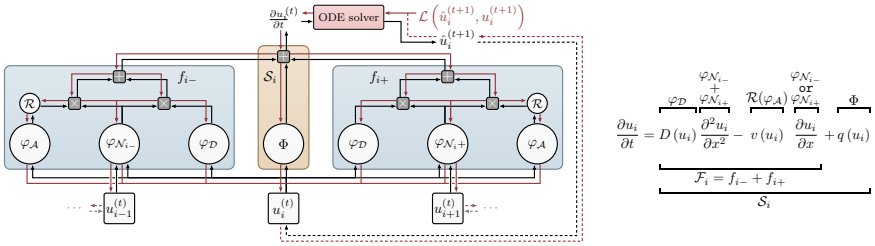


Figure 1: Flux (blue) and state (orange) kernels in FINN for the one-dimensional case (left)—red lines indicate gradient flow—and detailed assignment of the individual modules with their contribution to Eq. 1 (right).

Module $\varphi_{\mathcal{D}}$, receiving u_i as input, is responsible for the diffusive flux (the process of quantity homogenization from areas of high to low concentration). In case the diffusion coefficient D depends on u, the module is designed as feedforward network, such that $\varphi_{\mathcal{D}}(u) \approx D(u)$. Otherwise, $\varphi_{\mathcal{D}}$ is set as scalar value, which can be set trainable if the value of D is unknown.

Things are more complicated for advective flux, representing bulk motion and thus quantity that enters volume i either from left *or* right; module $\mathcal{R}$ decides on this based on the output of $\varphi_{\mathcal{A}}$ (which itself is a feedforward network, similarly to $\varphi_{\mathcal{D}}$). Technically, module $\mathcal{R}$ applies an upwind differencing scheme to prevent numerical instability in the first order spatial derivative calculation (Versteeg & Malalasekera, 1995) and is computed as

$$\mathcal{R}\left(\varphi_{\mathcal{A}}\left(u_i\right)\right) = \begin{cases} \mathrm{ReLU}\left(\varphi_{\mathcal{A}}\left(u_i\right)\right) & \text{on } f_{i-} \\ -\mathrm{ReLU}\left(-\varphi_{\mathcal{A}}\left(u_i\right)\right) & \text{on } f_{i+} \end{cases}. \tag{4}$$

It ensures that further processing of the advective flux from $\varphi_{\mathcal{A}}(u_i)$ is performed only on one control volume surface (either left or right), by switching on only the left surface element when $\varphi_{\mathcal{A}} > 0$ or only the right surface element when $\varphi_{\mathcal{A}} < 0$.

Finally, considering Eq. 4, the flux calculations turn into

$$f_{i-} = \varphi_{\mathcal{N}}(u_i, u_{i-1}) \cdot \left(\varphi_{\mathcal{D}}(u_i) + \mathcal{R}\big(\varphi_{\mathcal{A}}(u_i)\big)\right) \tag{5}$$

$$f_{i+} = \varphi_{\mathcal{N}}(u_i, u_{i+1}) \cdot \left(\varphi_{\mathcal{D}}(u_i) + \mathcal{R}\big(\varphi_{\mathcal{A}}(u_i)\big)\right) \tag{6}$$

$$\mathcal{F}_i = f_{i-} + f_{i+}. \tag{7}$$

Since the advective flux is only considered on one side of volume i (due to module $\mathcal{R}$), the summation of the numerical stencil from both f_{i-} and f_{i+} in Eq. 7 leads to $[-1, 1]$, being applied to $[u_i, u_{i-1}]$ when $\varphi_{\mathcal{A}} > 0$, or $[u_i, u_{i+1}]$ when $\varphi_{\mathcal{A}} < 0$ (i.e. only first order spatial derivative). For the diffusive flux calculation, on the other hand, the summation of the numerical stencil leads to the classical one-dimensional numerical Laplacian with $[1, -2, 1]$ applied to $[u_{i-1}, u_i, u_{i+1}]$, representing the second order spatial derivative, since the calculation of $\varphi_{\mathcal{D}}$ is performed on both surfaces (see subsection B.3 for a derivation). Altogether, module $\mathcal{R}$ generates the inductive bias to make $\varphi_{\mathcal{A}}$ only approximate the advective, and $\varphi_{\mathcal{D}}$ the diffusive flux.

Boundary Conditions A means of applying boundary conditions in a model is essential when solving PDEs. Currently available models mostly adopt convolution operations to model spatiotemporal processes. However, convolutions only allow a constant value to be padded at the domain boundaries (e.g. zero-padding or mirror-padding), which is only appropriate for the implementation of Dirichlet or periodic boundary condition types. Other types of frequently used boundary conditions are Neumann and Cauchy. These are defined as a derivative of the quantity of interest, and hence cannot be easily implemented in convolutional models. However, with certain pre-defined boundary condition types in FINN, the flux kernels at the boundaries are adjusted accordingly to allow for straightforward boundary condition consideration. For Dirichlet boundary condition, a constant value $u = u_b$ is set as the input u_{i-1} (for the flux kernel f_{i-}) or u_{i+1} (for f_{i+}) at the corresponding boundary. For Neumann boundary condition ν, the output of the flux kernel f_{i-} or f_{i+} at the corresponding boundary is set to be equal to ν. With Cauchy boundary condition, the solution-dependent derivative is calculated and set as u_{i-1} or u_{i+1} at the corresponding boundary.

State Kernel The state kernel $\mathcal{S}$ calculates the source/sink and storage terms of Eq. 1. The source/sink (if required) is learned using the module $\Phi(u) \approx q(u)$. The storage, $\frac{\partial u}{\partial t}$, is then calculated using the output of the flux kernel and module Φ of the state kernel:

$$\mathcal{S}_i = \mathcal{F}_i(u_{i-1}, u_i, u_{i+1}) + \Phi(u_i) \approx \frac{\partial u_i}{\partial t}. \tag{8}$$

By doing so, the PDE in Eq. 1 is now reduced to a system of coupled ordinary differential equations (ODEs), which are functions of u_{i-1}, u_i, u_{i+1}, and t. Thus, the solutions of the coupled ODE system can be computed via numerical integration over time. Since first order explicit approaches, such as the Euler method (Butcher, 2008), suffer from numerical instability (Courant et al., 1967; Isaacson & Keller, 1994), we employ the neural ordinary differential equation method (Neural ODE, Chen et al., 2018) to reduce numerical instability via the Runge–Kutta adaptive time-stepping strategy. The Neural ODE evaluates $\frac{\partial u_i}{\partial t}$ in form of FINN at an arbitrarily fine Δt and integrates FINN's output over time from t to $t+1$; the resulting u_i^{t+1} is fed back into the network as input in the next time step, until the last time step is reached. Thus, the entire training is performed in closed-loop, improving stability and accuracy of the prediction compared to networks trained with teacher forcing, i.e. one-step-ahead prediction (Praditia et al., 2020). The weight update is based on the mean squared error (MSE) as loss and realized by applying backpropagation through time (indicated by red arrows in Figure 1). In short, FINN takes only the initial condition u at time $t = 0$ and propagates the dynamics forward interactively with Neural ODE.

4. Experiments, Results, and Discussion

4.1. Synthetic Dataset

To demonstrate FINN's performance in comparison to other models, four different equations are considered as applications. First, the two-dimensional *Burgers' equation* (Basdevant et al., 1986) is chosen as a challenging function, as it is a non-linear PDE with $v(u) = u$ that could lead to a shock in the solution $u(x, y, t)$. Second, the *diffusion-sorption equation* (Nowak & Guthke, 2016) is selected with the non-linear retardation factor $R(u)$ as coefficient for the storage term, which contains a singularity $R(u) \to \infty$ for $u \to 0$ due to the parameter choice. Third, the two-dimensional Fitzhugh–Nagumo equation (Klaasen & Troy, 1984) as candidate for a *diffusion-reaction equation* (Turing, 1952) is selected, which is challenging because it consists of two non-linearly coupled PDEs to solve two main unknowns: the activator u_1 and the inhibitor u_2. Fourth, the Allen–Cahn equation with a cubic reaction term is chosen, leading to multiple jumps in the solution $u(x, t)$. Details on all four equations, data generation, and architecture designs can be found in Appendix C.

For each problem, three different datasets are generated (by conventional numerical simulation): *train*, used to train the models, in-distribution test (*in-dis-test*), being the train data simulated with a longer time span to test the models' generalization ability (extrapolation), and out-of-distribution test (*out-dis-test*). *Out-dis-test* data are used to test a trained ML model under conditions that are far away from training conditions, not only in terms of querying outputs for unseen inputs. Instead, *out-dis-test* data query outputs with regards to changes *not* captured by the inputs. These are changes that the ML tool per definition cannot be made aware of during training. In this work, they are represented by data generated with different initial or boundary condition, to test the generalization ability of the models outside the training distributions. FINN is trained and compared with both spatiotemporal deep learning models such as TCN, ConvLSTM, DISTANA and physics-aware neural network models such as PINN and PhyDNet. All models are trained with ten different random seeds using PyTorch's default weight initialization. Mean and standard deviation of the prediction errors are summarized in Table 1 for *train*, *in-dis-test* and *out-dis-test*. Details of each run are reported in the appendix. It is noteworthy that PINN cannot be tested on the *out-dis-test* dataset, since PINN assumes that the unknown variable u is an explicit function of x and t, and hence, when the initial or boundary condition is changed, the function will also be different and no longer valid.

4.1.1. Results

Burgers' The predictions of the best trained model of each method for the *in-dis-test* and the *out-dis-test* data are shown in Figure 2 and Figure 3, respectively. Both TCN and ConvLSTM fail to produce reasonable predictions, but qualitatively the other models manage to capture the shape of the data sufficiently, even towards the end of the *in-dis-test* period, where closed loop prediction has been applied for 380 time steps (after 20 steps of teacher forcing, see subsection C.1 for details). DISTANA, PINN, and FINN stand out in particular, but FINN produces more consistent and accurate predictions, evidenced by the mean value of the prediction errors. When tested against data generated with a different initial condition (*out-dis-test*), all models except for TCN and PhyDNet perform well. However, FINN still outperforms the other models with a significantly lower prediction error. The advective velocity learned by FINN's module $\varphi_{\mathcal{A}}$ is shown in Figure 5 (top left) and verifies that it successfully learned the advective velocity to be described by an identity function.

Diffusion-Sorption The predictions of the best trained model of each method for the concentration u from the diffusion-sorption equation are shown in Figure 12 of the appendix. TCN and ConvLSTM are shown to perform poorly even on the *train* data, evidenced by the high mean value of the prediction errors. On *in-dis-test* data, all models successfully produce relatively accurate predictions. However, when tested against different boundary conditions (*out-dis-test*), only FINN is able to capture the modifications and generalize well. The other models are shown to still overfit to the different boundary condition used in

Composing Partial Differential Equations with Physics-Aware Neural Networks

Table 1: Comparison of relative MSE (rMSE, which is the MSE divided by the variance) and standard deviation scores across ten repetitions between different deep learning (above dashed line) and physics-aware neural networks (below dashed line) on different equations. Best results reported in bold.

Equation	Model	Params	Dataset		
			Train	*In-dis-test*	*Out-dis-test*
Burgers' 2D	TCN	29 690	$(3.9 \pm 1.3) \times 10^{-2}$	$(3.1 \pm 0.8) \times 10^{-1}$	$(9.3 \pm 2.1) \times 10^{-2}$
	ConvLSTM	22 600	$(2.4 \pm 3.7) \times 10^{-1}$	$(6.0 \pm 6.7) \times 10^{0}$	$(4.2 \pm 3.5) \times 10^{-1}$
	DISTANA	19 100	$(7.3 \pm 11.0) \times 10^{-3}$	$(2.9 \pm 4.3) \times 10^{-1}$	$(3.6 \pm 3.6) \times 10^{-2}$
	PINN	3 041	$(1.8 \pm 0.8) \times 10^{-1}$	$(5.8 \pm 2.5) \times 10^{-1}$	—
	PhyDNet	185 300	$(1.1 \pm 0.3) \times 10^{-3}$	$(2.6 \pm 2.4) \times 10^{-1}$	$(3.5 \pm 1.5) \times 10^{-1}$
	FINN	**421**	$\mathbf{(1.0 \pm 0.6) \times 10^{-4}}$	$\mathbf{(1.1 \pm 0.4) \times 10^{-2}}$	$\mathbf{(2.4 \pm 1.0) \times 10^{-5}}$
Diffusion-sorption 1D	TCN	3 834	$(1.6 \pm 2.2) \times 10^{0}$	$(1.8 \pm 2.5) \times 10^{0}$	$(3.3 \pm 4.2) \times 10^{0}$
	ConvLSTM	3 960	$(5.1 \pm 4.6) \times 10^{-1}$	$(4.4 \pm 3.4) \times 10^{-1}$	$(1.8 \pm 1.2) \times 10^{0}$
	DISTANA	3 739	$(7.4 \pm 3.9) \times 10^{-4}$	$(3.5 \pm 3.6) \times 10^{-2}$	$(1.4 \pm 1.4) \times 10^{-1}$
	PINN	3 042	$(7.5 \pm 13.5) \times 10^{-4}$	$(6.1 \pm 12.8) \times 10^{-2}$	—
	PhyDNet	37 815	$\mathbf{(5.6 \pm 2.7) \times 10^{-4}}$	$(1.3 \pm 2.3) \times 10^{-1}$	$(5.0 \pm 2.8) \times 10^{-1}$
	FINN	**528**	$(7.6 \pm 7.4) \times 10^{-4}$	$\mathbf{(1.9 \pm 1.9) \times 10^{-3}}$	$\mathbf{(1.2 \pm 1.1) \times 10^{-3}}$
Diffusion-reaction 2D	TCN	31 734	$(1.9 \pm 1.2) \times 10^{-1}$	$(3.8 \pm 1.7) \times 10^{0}$	$(4.4 \pm 2.6) \times 10^{0}$
	ConvLSTM	24 440	$(1.2 \pm 2.8) \times 10^{-1}$	$(7.4 \pm 3.9) \times 10^{-1}$	$(4.4 \pm 3.8) \times 10^{-1}$
	DISTANA	75 629	$(5.3 \pm 4.5) \times 10^{-2}$	$(1.4 \pm 0.5) \times 10^{0}$	$(3.9 \pm 2.6) \times 10^{-1}$
	PINN	3 062	$(3.6 \pm 2.5) \times 10^{-3}$	$(5.6 \pm 5.0) \times 10^{-1}$	—
	PhyDNet	185 589	$\mathbf{(1.0 \pm 0.1) \times 10^{-3}}$	$(6.2 \pm 1.4) \times 10^{-1}$	$(1.0 \pm 0.4) \times 10^{0}$
	FINN	**882**	$(1.7 \pm 0.4) \times 10^{-3}$	$\mathbf{(1.6 \pm 0.4) \times 10^{-2}}$	$\mathbf{(1.8 \pm 0.1) \times 10^{-1}}$
Allen–Cahn 1D	TCN	10 052	$(4.4 \pm 9.1) \times 10^{-1}$	$(5.1 \pm 5.4) \times 10^{-1}$	$(2.8 \pm 3.8) \times 10^{-1}$
	ConvLSTM	7 600	$(2.8 \pm 5.3) \times 10^{-1}$	$(8.8 \pm 5.4) \times 10^{-1}$	$(4.0 \pm 4.5) \times 10^{-1}$
	DISTANA	6 422	$(2.3 \pm 1.0) \times 10^{-3}$	$(1.3 \pm 0.5) \times 10^{-1}$	$(5.4 \pm 3.5) \times 10^{-2}$
	PINN	3 021	$(9.7 \pm 6.2) \times 10^{-5}$	$(1.2 \pm 1.9) \times 10^{-1}$	—
	PhyDNet	37 718	$(4.6 \pm 3.0) \times 10^{-4}$	$(1.7 \pm 0.6) \times 10^{-1}$	$(7.5 \pm 2.1) \times 10^{-1}$
	FINN	**422**	$\mathbf{(3.2 \pm 3.5) \times 10^{-5}}$	$\mathbf{(3.5 \pm 4.2) \times 10^{-5}}$	$\mathbf{(3.6 \pm 4.5) \times 10^{-5}}$

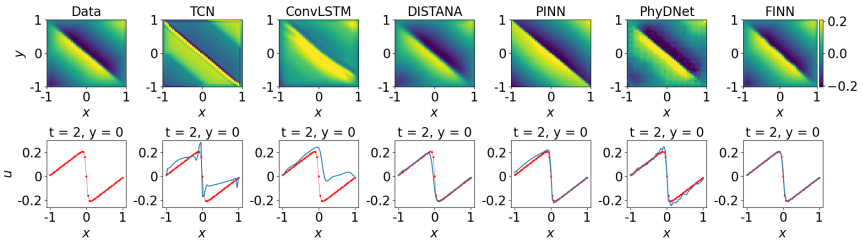


Figure 2: Plots of Burgers' data (red) and *in-dis-test* (blue) prediction using different models. The first row shows the solution over x and y at $t = 2$, and the second row visualizes the best model's solution distributed in x at $y = 0$ and $t = 2$.

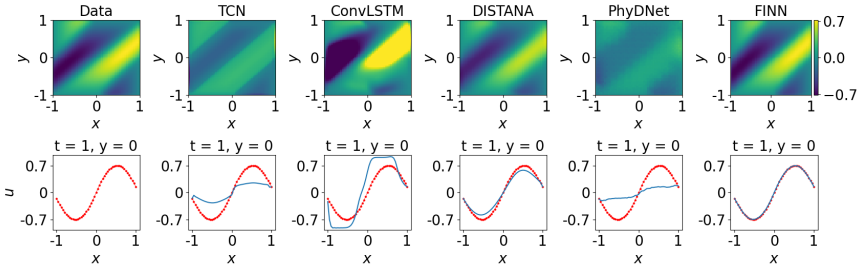


Figure 3: Plots of Burgers' data (red) and prediction (blue) of *out-dis-test* data using different models. The first row shows the solution over x and y at $t = 1$, and the second row visualizes the best model's solution over x at $y = 0$ and $t = 1$.

the train data (as detailed in subsection C.2). The retardation factor $R(u)$ learned by FINN's $\varphi_{\mathcal{D}}$ module is shown in Figure 5 (top center). The plot shows that the module learned the Freundlich retardation factor with reasonable accuracy.

Diffusion-Reaction The predictions of the best trained model of each method for activator u_1 in the diffusion-reaction equation are shown in Figure 15 (appendix) and Figure 4. TCN is again shown to fail to learn sufficiently from the *train* data. On *in-dis-test* data, DISTANA and the physics-aware models all predict with reasonable accuracy. When tested against data with different initial condition (*out-dis-test*), however, DISTANA and PhyDNet produce predictions with lower accuracy, and we find that FINN is the only model producing relatively low prediction errors. The reaction functions learned by FINN's Φ module are shown in Figure 5 (bottom). The plots show that the module successfully learned the Fitzhugh–Nagumo reaction function, both for the activator and inhibitor.

Allen–Cahn Results on the Allen–Cahn equation mostly align with those on Burgers' equation, confirming prior findings. However, it is worth noting that, again, only FINN is able to clearly represent the multiple nonlinearities caused by the exponent of third order in the reaction term, as visualized in Figure 16 and Figure 17 of the appendix. Again, Figure 5 (top right) shows that FINN accurately learned the dataset's reaction function.

4.1.2. Discussion

Overall, even with a high number of parameters, the prediction errors obtained using the pure ML methods (TCN, ConvLSTM, DISTANA) are worse compared to the physics-aware models, as shown in Table 1. As a physics-aware model, PhyDNet also possesses a high number of parameters. However, most of the parameters are allocated to the data-driven part (i.e. ConvLSTM branch), compared to the physics-aware part (i.e. PhyCell branch). In contrast to the other pure ML methods, DISTANA predicts with higher accuracy. This could act as an evidence that appropriate structural design of a neural network is as essential as physical knowledge to regularize the model training.

Among the physics-aware models, PINN and PhyDNet lie on different extremes. On one side, PINN requires complete knowledge of the modelled system in form of the equation. This means that all the functions, such as the advective velocity in Burgers', the retardation factor in the diffusion-sorption, and the reaction functions in the diffusion-reaction and Allen–Cahn equations, have to be pre-defined in order to train the model. This leaves less room for learning from data and could be error-prone if the designer's assumption is incorrect. On the other side, PhyDNet relies more heavily on the data-driven part and, therefore, could overfit the data. This can be shown by the fact that PhyDNet reaches the lowest training errors for the diffusion-sorption and diffusion-reaction equation predictions compared to the other models, but its performance significantly deteriorates when applied to *in-* and *out-dis-test* data. Our proposed model, FINN, lies somewhere in between these two extremes, compromising between putting sufficient physical knowledge into the model and leaving room for learning from data. As a consequence, we observe FINN showing excellent generalization ability. It outperforms other models up to multiple orders of magnitude, especially on *out-dis-test* data, when tested with different initial and boundary conditions; which is considered a particularly challenging task for ML models. Furthermore, the structure of FINN allows the extractions of learned functions such as the advective velocity, retardation factor, and reaction functions, showing good model interpretability.

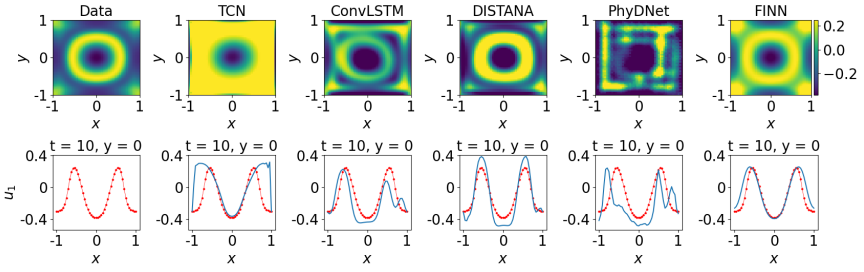


Figure 4: Plots of diffusion-reaction's activator data (red) and prediction (blue) of unseen dataset using different models. The plots in the first row show the solution distributed over x and y at $t = 10$, and the plots in the second row show the solution distributed in x at $y = 0$ and $t = 10$.

We also show that FINN properly handles the provided numerical boundary condition, as evidenced when applied to the test data that is generated with a different left boundary condition value, visualized in Figure 12 (appendix). Here, the test data is generated with a Dirichlet boundary condition $u(0, t) = 0.7$, which is lower than the value used in the train data, $u(0, t) = 1.0$. In fact, FINN is the only model that appropriately processes this different boundary condition value so that the prediction fits the test data nicely. The other models overestimate their prediction by consistently showing a tendency to still fit the prediction to the higher boundary condition value encountered during training.

Even though the spatial resolution used for the synthetic data generation is relatively coarse, leading to sparse data availability, FINN generalizes well. PINN, on the other hand, slightly suffers from the low resolution of the train data, although it still shows reasonable test performance.

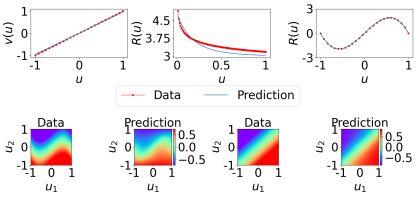


Figure 5: Plots of the learned functions (blue) as a function of u compared to the data (red) for Burgers' (top left), diffusion-sorption (top center), and Allen–Cahn (top right). The learned activator u_1 and inhibitor u_2 reaction functions in the diffusion-reaction equation are contrasted to the corresponding ground truth (second row).

Nevertheless, we conducted an experiment showing that PINN performs slightly better and more consistently when trained on higher resolution data (see appendix, subsection D.4), albeit still worse than FINN on coarse data. Therefore, we conclude that FINN is also applicable to real observation data that are often available only in low resolution, and/or in limited quantity. We demonstrate this further in subsection 4.2, when we apply FINN to real experimental data. FINN's generalization ability is superior to PINN, due to the fact that it is not possible to apply a trained PINN model to predict data with different initial or boundary condition. Additionally, we compare conservation errors on Burgers' for all models and find that FINN has the lowest with 7.12×10^{-3} (cf. PINN with 9.72×10^{-2}), even though PINN is explicitly trained to minimize conservation errors.

In terms of interpretability, FINN allows the extraction of functions learned by its dedicated modules. These learned functions can be compared with the real functions that generated the data (at least in the synthetic data case); examples are shown in Figure 5. The learned functions are the main data-driven discovery part of FINN and can also be used as a physical plausibility check. PhyDNet also comprises of a physics-aware and a data-driven part. However, it is difficult, if not impossible, to infer the learned equation from the model. Furthermore, the data-driven part of PhyDNet does not possess comparable interpretability, and could lead to overfitting as discussed earlier.

4.1.3. Ablations

In a number of ablation studies, we shed light on the relevance of particular choices in FINN: We substantiate the use of feedforward modules over polynomial regression in subsection D.1, proof FINN's ability to adequately learn unknown constituents under noisy conditions in subsec-

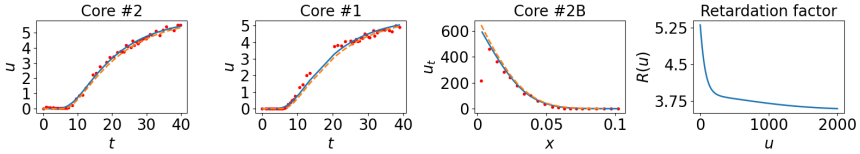


Figure 6: Breakthrough curve prediction of FINN (blue line) during training using data from core sample #2 (left), during testing using data from core sample #1 (second from left) and total concentration profile prediction using data from core sample #2B (second from right). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The right-most plot shows the learned retardation factor $R(u)$.

tion D.2, and consolidate the advantages of applying Neural ODE compared to traditional closed-loop application and Euler integration in subsection D.3.

4.2. Experimental Dataset

Real observation data are often available in limited amount, and they only provide partial information about the system of interest. To demonstrate how FINN can cope with real observation data, we use experimental data for the diffusion-sorption problem. The experimental data are collected from three different core samples that are taken from the same geographical area: #1,#2, and #2B (see subsection C.5 in the appendix for details). The objective of this experiment is to learn the retardation factor function from one of the core samples that concurrently applies to all the other samples. For this particular purpose, we only implement FINN since the other models have no means of learning the retardation factor explicitly. Here, we use the module $\varphi_{\mathcal{D}}$ to learn the retardation factor function, and we assume that the diffusion coefficient values of all the core samples are known. FINN is trained with the breakthrough curve of u, which is the dissolved concentration only at $x = L|_{0 \leq t \leq t_{\text{end}}}$ (i.e. only 55 data points).

Results and Discussion FINN reaches a higher accuracy for the training with core sample #2, with $\text{rMSE} = 3.38 \times 10^{-3}$ compared to a calibrated physical model from Nowak & Guthke (2016) with $\text{rMSE} = 7.42 \times 10^{-3}$, because the latter has to assume a specific function $R(u)$ with a few parameters for calibration. Our learned retardation factor is then applied and tested to core samples #1 and #2B. Figure 6 shows that FINN's prediction accuracy is competitive compared to the calibrated physical model. For core sample #1, FINN's prediction has an accuracy of 8.28×10^{-3} compared to the physical model that underestimates the breakthough curve (i.e. concentration profile) with $\text{rMSE} = 1.52 \times 10^{-2}$. Core sample #2B has significantly longer length than the other samples, and therefore a no-flow Neumann boundary condition was assumed at the bottom of the core. Because there is no breakthrough curve data available for this specific sample, we compare the prediction against the so-called total concentration profile $u(x, t_{\text{end}})$ at the end of the experiment. FINN produced a prediction with an accuracy of 6.39×10^{-2}, whereas the physical model overestimates the concentration with $\text{rMSE} = 1.50 \times 10^{-1}$. To briefly summarize, FINN is able to learn the retardation factor from a sparse experimental dataset and apply it to other core samples with similar soil properties with reasonable accuracy, even with a different boundary condition type.

5. Conclusion

Spatiotemporal dynamics often can be described by means of advection-diffusion type equations, such as Burgers', diffusion-sorption, or diffusion-reaction equations. When modeling those dynamics with ANNs, large efforts must be taken to prevent the models from overfitting (given the model is able to learn the problem at all). The incorporation of physical knowledge as regularization yields robust predictions beyond training data distributions.

With FINN, we have introduced a modular, physics-aware ANN with excellent generalization abilities beyond different initial and boundary conditions, when contrasted to pure ML models and other physics-aware models. FINN is able to model and extract unknown constituents of differential equations, allowing high interpretability and an assessment of the plausibility of the model's out-of-distribution predictions. As next steps we seek to apply FINN beyond second order spatial derivatives, improve its scalability to large datasets, and make it applicable to heterogeneously distributed data (i.e. represented as graphs) by modifying the module $\varphi_{\mathcal{N}}$ to approximate variable and location-specific stencils (for more details on limitations of FINN, the reader is referred to subsection B.2). Another promising future direction is the application of FINN to real-world sea-surface temperature and weather data.

Software and Data

Code and data that are used for this paper can be found in the repository https://github.com/CognitiveModeling/finn.

Acknowledgements

We thank the reviewers for constructive feedback, leading us to evaluate all methods on two-dimensional instead of one-dimensional Burgers', to additionally benchmark APHINITY (Yin et al., 2020), FNO (Li et al., 2020a), and cvPINN (Patel et al., 2022), and to perform a CNN-NODE ablation study. Results are reported in Figure 7 and Table 2 and discussed in the according sections.

This work is funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016 as well as EXC 2064 – 390727645. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). Moreover, we thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer.

References

Bar-Sinai, Y., Hoyer, S., Hickey, J., and Brenner, M. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Basdevant, C., Deville, M., Haldenwang, P., Lacroix, J., Ouazzani, J., Peyret, R., Orlandi, P., and Patera, A. Spectral and finite difference solutions of the burgers equation. *Computers & Fluids*, 14(1):23–41, 1986. doi: https://doi.org/10.1016/0045-7930(86)90036-8.

Battaglia, P., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Butcher, J. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008. ISBN 9780470753750.

Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Courant, R., Friedrichs, K., and Lewy, H. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. doi: 10.1147/rd.112.0215.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Fornberg, B. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

Guen, V. and Thome, N. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11474–11484, 2020.

Isaacson, E. and Keller, H. *Analysis of Numerical Methods*. Dover Books on Mathematics. Dover Publications, 1994. ISBN 9780486680293.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A., Graves, A., and Kavukcuoglu, K. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., and Butz, M. A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint arXiv:1912.11141*, 2019.

Klaasen, G. and Troy, W. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1):96–110, 1984. doi: 10.1137/0144008.

Kochkov, D., Smith, J., Alieva, A., Wang, Q., Brenner, M., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

Lake, B. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems 32*, pp. 9791–9801. 2019.

Lake, B., Ullman, T., Tenenbaum, J., and Gershman, S. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 2017. doi: 10.1017/S0140525X16001837.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Long, Z., Lu, Y., Ma, X., and Dong, B. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018.

Moukalled, F., Mangani, L., and Darwish, M. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. doi: 10.1007/978-3-319-16874-6.

Nowak, W. and Guthke, A. Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, 18(11), 2016. doi: 10.3390/e18110409.

Patel, R. G., Manickam, I., Trask, N. A., Wood, M. A., Lee, M., Tomas, I., and Cyr, E. C. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *Journal of Computational Physics*, 449: 110754, 2022.

Praditia, T., Walser, T., Oladyshkin, S., and Nowak, W. Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, 13(15):3873, 2020. doi: 10.3390/en13153873.

Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M., and Nowak, W. Finite volume neural network: Modeling subsurface contaminant transport. *arXiv preprint arXiv:2104.06010*, 2021.

Raissi, M., Perdikaris, P., and Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Riley, K., Hobson, M., and Bence, S. *Mathematical Methods for Physics and Engineering*. Cambridge University Press, 2009. ISBN 9780521139878.

Seo, S., Meng, C., and Liu, Y. Physics-aware difference graph networks for sparsely-observed dynamics. In *International Conference on Learning Representations*, 2019.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Smolarkiewicz, P. A simple positive definite advection scheme with small implicit diffusion. *Monthly Weather Review*, 111(3):479 – 486, 1983. doi: 10.1175/1520-0493.

Turing, A. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

Versteeg, H. and Malalasekera, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. New York, 1995. ISBN 9780470235157.

Yin, Y., Guen, V., Dona, J., Ayed, I., de Bézenac, E., Thome, N., and Gallinari, P. Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*, 2020.

Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M., and Hoyer, S. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605, 2021.

A. Appendix

This appendix is structured as follows: Additional detailed differences between FINN and the benchmark models used in this work are presented in subsection B.1. Then, limitations of FINN are discussed briefly in subsection B.2. Detailed numerical derivation of the first and second order spatial derivative is shown in subsection B.3. Additional details on the equations used as well as model specifications and in-depth results are presented in subsection C.1 (Burgers'), subsection C.2 (diffusion-sorption), subsection C.3 (diffusion-reaction), and subsection C.4 (Allen–Cahn). Moreover, the soil parameters and simulation domain used in the diffusion-sorption laboratory experiment are presented in subsection C.5. The results of our ablation studies are reported in Appendix D: Additional polynomial-fitting baselines to account for the equations are outlined in subsection D.1, including an ablation where we replace the neural network modules of FINN by polynomials. A robustness analysis of FINN can be found in subsection D.2, where our method is evaluated on noisy data from all equations. The role of the Neural ODE module is evaluated in subsection D.3, accompanied with a runtime analysis. Finally, results of training PINN with high-resolution data and training PhyDNet with the original architecture (more parameters) are reported in subsection D.4 and subsection D.5, respectively.

B. Methodological Supplements

B.1. Distinction to Related Work

Pure ML models Originally, FINN was inspired by DISTANA, a pure ML model proposed by Karlbauer et al. (2019). While DISTANA has large similarities to Shi et al. (2015)'s ConvLSTM, it propagates lateral information through an additional latent map—instead of reading lateral information from the input map directly, as done in ConvLSTM—and aggregates this lateral information by means of a user-defined combination of arbitrary layers. Accordingly, the processing of the two-point flux approximation in FINN, using the lateral information, is more akin to DISTANA, albeit motivated and augmented by physical knowledge. As a result, the lateral information flow is guaranteed to behave in a physically plausible manner. That is, quantity can either be locally generated by a source (increased), absorbed by a sink (decreased), or spatially distributed (move to neighboring cells).

Terminology separates the spatially distribution of quantity into diffusion and advection. Diffusion describes the equalization of quantity from high to low concentration levels, whereas advection is defined as the bulk motion of a large group of particles/atoms caused by external forces. FINN ensures the conservation of laterally propagating quantity, i.e. what flows from left to right will effectively cause a decrease of quantity at the left and an increase at the right. Pure ML models (without physical constraints) cannot be guaranteed to adhere to these fundamental rules.

PINN Since ML models guided by physical knowledge not only behave empirically plausible but also require fewer parameters and generalize to much broader extent, numerous approaches have been proposed recently to combine artificial neural networks with physical knowledge. Raissi et al. (2019) introduced the physics-informed neural network (PINN), a concrete and outstanding model that explicitly learns a provided PDE. As a result, PINN mimics e.g. Burgers' equation (see Eq. 15) by learning the quantity function $u(x, y, t)$ for defined initial and boundary conditions with a feedforward network. The partial derivatives are implicitly realized by automatic differentiation of the feedforward network (representing u) with respect to the input variables x, y, or t. The learned neural network thus satisfies the constraints defined by the partial derivatives that are formulated in the loss term.

Accordingly, PINN has the advantage to explicitly provide a (continuous) solution of the desired function $u(x, y, t)$ and, correspondingly, predictions can be queried for an arbitrary combination of input values, circumventing the need for simulating the entire domain with e.g. a carefully chosen simulation step size. However, this advantage prevents a trained PINN from being applied to data with different initial and boundary conditions than those of the training data; a limitation that still holds for successors such as the recently proposed control volume discretization PINN (cvPINN, Patel et al., 2022) that is reported to behave well on challenging shock problems. While cvPINN is able to maintain the effect of a particular boundary condition, it still cannot generalize to unseen boundary conditions, as outlined in Figure 7. In contrast, FINN does not learn the explicit function $u(x, y, t)$ defined for particular initial and boundary conditions, but approximates the distinct components of the function. These components are combined—as suggested by the physical equation—to result in a compositional model that is more universally applicable. That is, in stark contrast to PINN, the compositional function learned by FINN can be applied to varying initial and boundary conditions, since the learned individual components provide the same functionality as the corresponding components of the PDE when processed by a numerical solver. Applying the conventional numerical solver (e.g. FVM) would require either complete knowledge of the equation or careful calibration of the unknowns by choosing equations from a library of possible solutions and tuning the parameters. But FINN's data driven component can reveal unknown relations, such as the retardation factor of a function, without the need for subjective prior assumptions.

Composing Partial Differential Equations with Physics-Aware Neural Networks

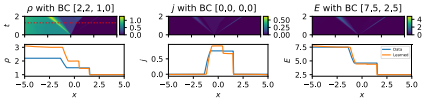


Figure 7: Results of cvPINN with left BC of ρ changed from 3.0 to 2.2. Top: MSE between ground truth and prediction. Bottom: u states in timestep $t = 1.0$ (step 128 in simulation). Code adapted from https://github.com/rgp62/cvpinns.

Thus, FINN paves the way for non-experts in numerical simulation to accurately model physical phenomena by learning unknown constituents on the fly that otherwise require both sophisticated domain knowledge and intensive analysis of the problem at hand.

Learning Derivatives via Convolution An alternative and much addressed approach of implanting physical plausibilty into artificial neural networks is to implicitly learn the first n derivatives using appropriately designed convolution kernels (Long et al., 2018; Li et al., 2020a;b; Guen & Thome, 2020; Yin et al., 2020; Sitzmann et al., 2020). These methods exploit the link that most PDEs, such as $u(t, x, y, \dots)$, can be reformulated as

$$\frac{\partial u}{\partial t} = F\left(t, x, y, \dots, u, \frac{\partial u}{\partial x}, \frac{\partial u}{y}, \frac{\partial u}{\partial x \partial y}, \frac{\partial^2 u}{\partial x^2}, \dots\right). \tag{9}$$

When learning partial derivatives up to order n and setting irrelevant features to zero, these methods have, in principle, the capacity to represent most PDEs. However, the degrees of freedom in these methods are still very high and can fail to safely guide the ML algorithm. FINN accounts for the first and second derivative by learning the corresponding stencil in the module $\varphi_{\mathcal{N}}$ and combining this stencil with the case-sensitive ReLU module $\mathcal{R}$, which allows a precise control of the information flow resulting from the first and second derivative. More importantly, convolutions only allows implementation of Dirichlet and periodic boundary conditions (by means of zero- or mirror-padding), and is not appropriate for implementing other boundary condition types.

Learning ODE Coefficients The group around Brenner and Hoyer (Bar-Sinai et al., 2019; Kochkov et al., 2021; Zhuang et al., 2021) follow another line of research about learning the coefficients of ordinary differential equations (ODEs); note that PDEs can be transformed into a set of coupled ODEs by means of polynomial expansion or spectral differentiation as shown in Bar-Sinai et al. (2019). The physical constraint in these works is mostly realized by satisfying the temporal derivative as loss.

While this approach shares similarities with our method by incorporating physically motivated inductive bias into the model, it uses this bias mainly to improve interpolation from coarse to finer grid resolutions and thus to accelerate simulation. Our work focuses on discovering unknown relationships/laws (or re-discovering laws in the case of the synthetic examples), such as the advective velocity in the Burgers' example, the retardation factor function in the diffusion-sorption example, and the reaction function in the diffusion-reaction and Allen–Cahn example. Additionally, the works by Brenner and Hoyer employ a convolutional structure, which is only applicable to Dirichlet or periodic boundary conditions, and it suffers from a slight instability during training when the training data trajectory is unrolled for a longer period. In contrast, FINN employs the flux kernel, calculated at all control volume surfaces, which enables the implementation and discovery of various boundary conditions. Furthermore and in contrast to Brenner and Hoyer, FINN employs the Neural ODE method as the time integrator to reduce numerical instability during training with long time series. In accord with this line of research, FINN is also able to generalize well when trained with a relatively sparse dataset (coarse resolution), reducing the computational burden.

Numerical ODE Solvers Traditional numerical solvers for ordinary differential equations (ODEs) can be seen as the pure-physics contrast to FINN. However, these do not have a learning capacity to reveal unknown dependencies or functions from data. Also, as shown in (Yin et al., 2020), a pure Neural ODE without physical inductive bias does not reach the same level of accuracy as physics-aware neural network models. We confirm this finding with an ablation (see subsection D.3), where FINN is replaced by a CNN whose output is processed by Neural ODE (NODE), reaching worse results. Furthermore, the relevance of the data-driven component in FINN is underlined by our field experiment, where FINN reached lower errors on a real-world dataset compared to a conventional FVM model.

APHYNITY and FNO We also have conducted another series of experiments to quantitatively compare two additional state-of-the-art methods. APHYNITY is a physics-aware model that consists of a physics-part that can be augmented by a neural network component to compensate phenomena that are not covered by the physics-part. It was developed by Yin et al. (2020), the same group that has developed PhyDNet before. On the other hand, the Fourier neural operator (FNO) model, introduced by Li et al. (2020a), is a pure ML model that learns concrete functions of space and time in frequency domain, e.g. $u = f(x, t)$. This is similar to e.g. PINN and SIREN (Raissi et al., 2019; Sitzmann et al., 2020) and results in a continuous representation of the trained simulation domain for any input combi-

Composing Partial Differential Equations with Physics-Aware Neural Networks

Table 2: Comparison of rMSE and standard deviation scores across ten repetitions between CNN-NODE, FNO, APHYNITY, and FINN on different equations. Best results reported in bold.

Equation	Model	Params	Dataset		
			Train	*In-dis-test*	*Out-dis-test*
Burgers' 2D	CNN-NODE	2 657	$(2.0 \pm 0.7) \times 10^{-3}$	$(4.7 \pm 6.5) \times 10^{-1}$	$(9.1 \pm 9.5) \times 10^{-1}$
	FNO	116 115	$(6.4 \pm 9.2) \times 10^{-4}$	$(5.9 \pm 2.6) \times 10^{-2}$	$(9.7 \pm 0.9) \times 10^{-1}$
	APHYNITY	2 658	$(1.0 \pm 0.3) \times 10^{-2}$	$(4.3 \pm 0.1) \times 10^{-2}$	$(4.4 \pm 0.5) \times 10^{-4}$
	FINN	**421**	$\mathbf{(1.0 \pm 0.6) \times 10^{-4}}$	$\mathbf{(1.1 \pm 0.4) \times 10^{-2}}$	$\mathbf{(2.4 \pm 1.0) \times 10^{-5}}$
Diffusion-sorption 1D	CNN-NODE	1 026	$(6.8 \pm 14.3) \times 10^{-2}$	$(3.8 \pm 7.7) \times 10^{0}$	$(6.6 \pm 12.9) \times 10^{0}$
	FNO	5 878	$(1.1 \pm 2.8) \times 10^{-2}$	$(2.1 \pm 3.9) \times 10^{-2}$	$(4.1 \pm 1.4) \times 10^{-1}$
	APHYNITY	—	—	—	—
	FINN	**528**	$\mathbf{(7.6 \pm 7.4) \times 10^{-4}}$	$\mathbf{(1.9 \pm 1.9) \times 10^{-3}}$	$\mathbf{(1.2 \pm 1.1) \times 10^{-3}}$
Diffusion-reaction 2D	CNN-NODE	2 946	$(5.1 \pm 0.8) \times 10^{-1}$	$(2.3 \pm 0.9) \times 10^{1}$	$(3.3 \pm 0.8) \times 10^{0}$
	FNO	116 288	$(1.3 \pm 0.5) \times 10^{-4}$	$(4.8 \pm 11.0) \times 10^{1}$	$(2.9 \pm 1.0) \times 10^{0}$
	APHYNITY	2 949	$(2.9 \pm 0.5) \times 10^{-3}$	$(4.0 \pm 1.2) \times 10^{-1}$	$\mathbf{(8.0 \pm 1.8) \times 10^{-2}}$
	FINN	**882**	$\mathbf{(1.7 \pm 0.4) \times 10^{-3}}$	$\mathbf{(1.6 \pm 0.4) \times 10^{-2}}$	$(1.8 \pm 0.1) \times 10^{-1}$
Allen-Cahn 1D	CNN-NODE	929	$(4.2 \pm 2.6) \times 10^{-5}$	$(7.7 \pm 3.2) \times 10^{-2}$	$(8.1 \pm 5.4) \times 10^{-1}$
	FNO	5 745	$(3.3 \pm 3.2) \times 10^{-3}$	$(1.2 \pm 0.2) \times 10^{-1}$	$(1.5 \pm 0.1) \times 10^{0}$
	APHYNITY	930	$(1.2 \pm 0.0) \times 10^{-4}$	$(2.7 \pm 0.0) \times 10^{-3}$	$(6.7 \pm 0.1) \times 10^{-4}$
	FINN	**422**	$\mathbf{(3.2 \pm 3.5) \times 10^{-5}}$	$\mathbf{(3.5 \pm 4.2) \times 10^{-5}}$	$\mathbf{(3.6 \pm 4.5) \times 10^{-5}}$

nation of x and t.

Results are reported in Table 2 and support our previous findings, further fostering and demonstrating FINN's superior performance. Indeed, FNO and APHYNITY perform very well (FNO on extrapolation, APHYNITY on generalization), but they do not reach FINN's accuracy. Note that for the diffusion-reaction equation, APHYNITY outperforms FINN when tested with the *out-dis-test* data. This can be attributed to the fact that APHYNITY is trained assuming complete knowledge of the modelled equation. When parts of the modelled equation are unknown, APHINITY performs worse than FINN. This demonstrates that APHYNITY is very accurate when the full correct equation is known, whereas FINN can learn unknown parts of the equation better. Additionally, APHINITY fails to be trained with diffusion-sorption data due to instability issues.

B.2. Limitations of FINN

First, while the largest limitation of our current method can be seen in the capacity to only represent first and second order spatial derivatives, this is an issue that we will address in follow up work. Still, FINN can already be applied to a very wide range of problems as most equations in fact only depend on up to the second spatial derivative. *Second*, FINN is only applicable to spatially homogeneously distributed data so far—we intend to extend it to heterogeneous data on graphs. *Third*, although we have successfully applied FINN to two-dimensional Burgers' and diffusion-reaction data, the training time is considerable, albeit comparable to other physics-aware ANNs. While, according to our observations, this appears to be a common issue for physics-aware neural networks (cf. Table 17), an optimized implementation on today's hardware-accelerated computation could potentially widen the bottleneck.

B.3. Learning the Numerical Stencil

Semantically, the $\varphi_{\mathcal{N}}$ module learns the numerical stencil, that is the geometrical arrangement of a group of neighboring cells to approximate the derivative numerically, effectively learning the first spatial derivative $\frac{\partial u}{\partial x}$ from both $[u_{i-1}, u_i]$ and $[u_i, u_{i+1}]$, which are the inputs to the $\varphi_{\mathcal{N}}$ module.

The lateral information flowing from u_{i-1} and u_{i+1} toward u_i is controlled by the $\varphi_{\mathcal{A}}$ (advective flux, i.e. bulk motion of many particles/atoms that can either move to the left or to the right) and the $\varphi_{\mathcal{D}}$ (diffusive flux, i.e. drive of particles/atoms to equilibrium from regions of high to low concentration) modules. Since the advective flux can only move either to the left or to the right, it will be considered only in the left flux kernel (f_{i-}) or in the right flux kernel (f_{i+}), and not both at the same time. The case-sensitive ReLU module $\mathcal{R}$ (Eq. 4) decides on this, by setting the advective flux in the irrelevant flux kernel to zero (effectively depending on the sign of the output of $\varphi_{\mathcal{A}}$). Thus, the advective flux is only considered from either u_{i-1} or u_{i+1} to

u_i, which amounts to the first order spatial derivative.

The diffusive flux, on the other hand, can propagate from both sides towards the control volume of interest u_i and, hence, the second order spatial derivative, accounting for the difference between u_{i-1} and u_{i+1}, has to be applied. In our method, this is realized through the combination of the $\varphi_\mathcal{N}$ and $\varphi_\mathcal{D}$ modules, calculating the diffusive fluxes $\delta_- = \varphi_\mathcal{N}(u_i, u_{i-1})\varphi_\mathcal{D}(u_i)$ and $\delta_+ = \varphi_\mathcal{N}(u_i, u_{i+1})\varphi_\mathcal{D}(u_i)$ inside of the respective left and right flux kernel. The combination of these two deltas in the state kernel ensures the consideration of the diffusive fluxes from left (including u_{i-1}) and from right (including u_{i+1}), resulting in the ability to account for the second order spatial derivative.

Technically, the coefficients for the first, i.e. $[-1, 1]$, and second, i.e. $[1, -2, 1]$, order spatial differentiation schemes are common definitions and a derivation can be found, for example, in Fornberg (1988). However, a quick derivation of the Laplace scheme $[1, -2, 1]$ can be formulated as follows. Define the second order spatial derivative as the difference between two first order spatial derivatives, i.e.

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{(\partial u/\partial x)|_{i-} - (\partial u/\partial x)|_{i+}}{\Delta x} \tag{10}$$

with the two first order spatial derivatives—representing the differences between u_{i-1}, u_i (left, i.e. 'minus') and u_i, u_{i+1} (right, i.e. 'plus')—defined as

$$(\partial u/\partial x)|_{i-} \approx (u_{i-1} - u_i)/\Delta x \tag{11}$$

$$(\partial u/\partial x)|_{i+} \approx (u_i - u_{i+1})/\Delta x. \tag{12}$$

Then, substituting Eq. 11 and Eq. 12 into Eq. 10, we get

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{(u_{i-1} - u_i) - (u_i - u_{i+1})}{\Delta x^2} \tag{13}$$

$$\approx \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}, \tag{14}$$

hence the $[+1, -2, +1]$ as coefficients in the second order spatial derivative. In fact, in the Burgers' experiment, we find that FINN learns the appropriate stencils, i.e. $[-0.99 \pm 0.04, 0.99 \pm 0.04]$ for the first order spatial derivative.

C. Data and Model Details

C.1. Burgers'

The Burgers' equation is commonly employed in various research areas, including fluid mechanics.

Data The two-dimensional generalized Burgers' equation is written as

$$\frac{\partial u}{\partial t} = -v(u)\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) + D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{15}$$

where the main unknown is u, the advective velocity is denoted as $v(u)$ which is a function of u and the diffusion coefficient is $D = 0.01/\pi$. In the current work, the advective velocity function is chosen to be an identity function $v(u) = u$ to reproduce the experiment conducted in the PINN paper (Raissi et al., 2019).

The simulation domain for the ***train* data** is defined with $x = [-1, 1]$, $y = [-1, 1]$, $t = [0, 1]$ and is discretized with $N_x = 49$ and $N_y = 49$ spatial locations, and $N_t = 201$ simulation steps. The initial condition was defined as $u(x, y, 0) = -\sin(\pi(x + y))$, and the corresponding boundary condition was defined as $u(-1, y, t) = u(1, y, t) = u(x, -1, t) = u(x, 1, t) = 0$.

The ***in-dis-test* data** is simulated with with $x = [-1, 1]$, $y = [-1, 1]$ and with the time span of $t = [1, 2]$ and $N_t = 201$. Initial condition is taken from the train data at $t = 1$ and boundary conditions are also identical to the train data.

The simulation domain for the ***out-dis-test* data** was identical with the train data, except for the initial condition that was defined as $u(x, y, 0) = -\sin(\pi(x - y))$.

Model Architectures **TCN** is designed to have one input- and output neuron, and one hidden layer of size 32. **ConvLSTM** has one input- and output neuron and one hidden layer of size 24. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and a hidden layer of size 16 is used. The pure ML models were trained on the first 150 time steps and validated on the remaining 50 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** is defined as a feedforward network with the size of [3, 20, 20, 20, 20, 20, 20, 20, 20, 1]. **PhyDNet** is defined with the PhyCell containing 32 input dimensions, 49 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_\mathcal{N}$, $\varphi_\mathcal{D}$, $\mathcal{R}$ and $\varphi_\mathcal{A}$ were used, with $\varphi_\mathcal{A}$ defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes u as an input and outputs the advective velocity $v(u)$, and $\varphi_\mathcal{D}$ as a learnable scalar that learns the diffusion coefficient D. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 3), *in-dis-test* (Table 4 and Figure 8), and *out-dis-test* (Table 5 and Figure 9) datasets.

Table 3: Closed loop rMSE on the *train* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	6.2×10^{-2}	3.5×10^{-1}	3.6×10^{-3}	4.7×10^{-2}	7.9×10^{-4}	9.9×10^{-5}
2	2.5×10^{-2}	4.1×10^{-2}	7.1×10^{-3}	3.1×10^{-1}	8.6×10^{-4}	8.9×10^{-5}
3	3.9×10^{-2}	5.8×10^{-2}	3.5×10^{-4}	1.7×10^{-1}	1.3×10^{-3}	2.1×10^{-4}
4	3.7×10^{-2}	1.9×10^{-2}	4.5×10^{-3}	2.4×10^{-1}	7.9×10^{-4}	3.7×10^{-5}
5	5.7×10^{-2}	2.6×10^{-2}	1.2×10^{-3}	1.9×10^{-1}	8.8×10^{-4}	5.5×10^{-5}
6	2.5×10^{-2}	5.3×10^{-1}	4.0×10^{-3}	1.2×10^{-1}	1.7×10^{-3}	4.2×10^{-5}
7	4.1×10^{-2}	1.2×10^{0}	2.7×10^{-4}	1.3×10^{-1}	1.1×10^{-3}	1.3×10^{-4}
8	2.2×10^{-2}	2.4×10^{-2}	3.9×10^{-2}	1.7×10^{-1}	1.5×10^{-3}	1.2×10^{-4}
9	4.6×10^{-2}	7.8×10^{-2}	9.2×10^{-3}	2.8×10^{-1}	6.3×10^{-4}	1.9×10^{-4}
10	3.2×10^{-2}	2.6×10^{-2}	3.9×10^{-3}	1.2×10^{-1}	9.1×10^{-4}	6.2×10^{-5}

Table 4: Closed loop rMSE on *in-dis-test* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	4.1×10^{-1}	1.4×10^{1}	1.8×10^{-1}	7.0×10^{-2}	9.0×10^{-1}	1.0×10^{-2}
2	3.0×10^{-1}	2.6×10^{0}	2.4×10^{-1}	8.6×10^{-1}	2.9×10^{-1}	1.3×10^{-2}
3	2.2×10^{-1}	8.5×10^{-1}	1.5×10^{-2}	9.5×10^{-1}	3.6×10^{-1}	1.4×10^{-2}
4	2.9×10^{-1}	3.9×10^{-1}	3.5×10^{-1}	6.4×10^{-1}	1.5×10^{-1}	3.0×10^{-3}
5	3.5×10^{-1}	4.6×10^{-1}	2.9×10^{-2}	3.6×10^{-1}	1.5×10^{-1}	1.0×10^{-2}
6	3.4×10^{-1}	1.4×10^{1}	4.3×10^{-2}	6.8×10^{-1}	2.4×10^{-1}	7.5×10^{-3}
7	4.4×10^{-1}	1.9×10^{1}	1.5×10^{-2}	4.4×10^{-1}	2.0×10^{-1}	1.6×10^{-2}
8	2.2×10^{-1}	5.4×10^{-1}	1.5×10^{0}	5.3×10^{-1}	2.0×10^{-1}	1.2×10^{-2}
9	3.5×10^{-1}	4.7×10^{0}	4.6×10^{-1}	8.1×10^{-1}	1.1×10^{-2}	1.2×10^{-2}
10	1.9×10^{-1}	3.1×10^{0}	2.9×10^{-2}	4.4×10^{-1}	7.2×10^{-2}	1.1×10^{-2}

Table 5: Closed loop rMSE on *out-dis-test* data from ten different training runs for each model for the Burgers' equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.1×10^{-1}	3.4×10^{-1}	5.5×10^{-2}	-	3.8×10^{-1}	2.0×10^{-5}
2	8.4×10^{-2}	1.1×10^{0}	4.9×10^{-2}	-	7.0×10^{-1}	3.0×10^{-5}
3	1.2×10^{-1}	3.1×10^{-1}	3.5×10^{-3}	-	3.5×10^{-1}	2.9×10^{-5}
4	1.1×10^{-1}	1.3×10^{-1}	4.3×10^{-2}	-	3.0×10^{-1}	2.7×10^{-5}
5	1.0×10^{-1}	1.1×10^{-1}	1.3×10^{-2}	-	4.8×10^{-1}	8.4×10^{-6}
6	7.0×10^{-2}	5.8×10^{-1}	9.0×10^{-3}	-	1.9×10^{-1}	1.2×10^{-5}
7	8.2×10^{-2}	1.1×10^{0}	9.8×10^{-3}	-	4.0×10^{-1}	3.7×10^{-5}
8	4.7×10^{-2}	9.8×10^{-2}	1.3×10^{-1}	-	2.3×10^{-1}	2.5×10^{-5}
9	1.1×10^{-1}	1.8×10^{-1}	3.7×10^{-2}	-	1.6×10^{-1}	3.8×10^{-5}
10	9.5×10^{-2}	2.9×10^{-1}	1.4×10^{-2}	-	3.4×10^{-1}	1.6×10^{-5}

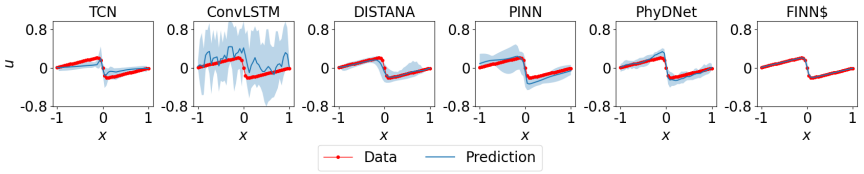


Figure 8: Prediction mean over ten different trained models (with 95% confidence interval) of the Burgers' equation at $t = 2$ for the *in-dis-test* dataset.

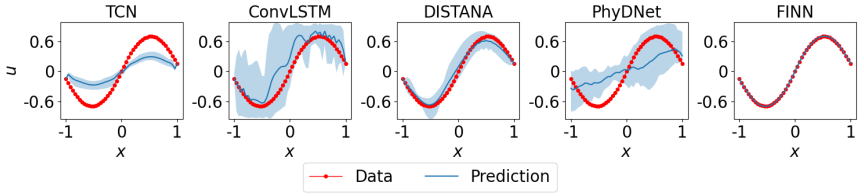


Figure 9: Prediction mean over ten different trained models (with 95% confidence interval) of the Burgers' equation at $t = 1$ for the *out-dis-test* data.

C.2. Diffusion-Sorption

The diffusion-sorption equation is another widely applied equation in fluid mechanics. A practical example of the equation is to model contaminant transport in groundwater. Its retardation factor R can be modelled using different closed parametric relations known as sorption isotherms that should be calibrated to observation data.

Data The one-dimensional diffusion-sorption equation is written as the following coupled system of equations:

$$\frac{\partial u}{\partial t} = \frac{D}{R(u)} \frac{\partial^2 u}{\partial x^2}, \quad (16) \qquad \frac{\partial u_t}{\partial t} = D\phi \frac{\partial^2 u}{\partial x^2}, \quad (17)$$

where the dissolved concentration u and total concentration u_t are the main unknowns. The effective diffusion coefficient is denoted by $D = 5 \times 10^{-4}$, the retardation factor is $R(u)$, a function of u, and the porosity is denoted by $\phi = 0.29$.

In this work, the Freundlich sorption isotherm, see details in (Nowak & Guthke, 2016), was chosen to define the retardation factor:

$$R(u) = 1 + \frac{1-\phi}{\phi} \rho_s k n_f u^{n_f - 1}, \quad (18)$$

where $\rho_s = 2\,880$ is the bulk density, $k = 3.5 \times 10^{-4}$ is the Freundlich's parameter, and $n_f = 0.874$ is the Freundlich's exponent.

The simulation domain for the ***train* data** is defined with $x = [0, 1]$, $t = [0, 2\,500]$ and is discretized with $N_x = 26$ spatial locations, and $N_t = 501$ simulation steps. The initial condition is defined as $u(x, 0) = 0$, and the boundary condition is defined as $u(0, t) = 1.0$ and $u(1, t) = D\frac{\partial u}{\partial x}$.

The ***in-dis-test* data** was simulated with $x = [0, 1]$ and with the time span of $t = [2\,500, 10\,000]$ and $N_t = 1\,501$. Initial condition is taken from the train data at $t = 2\,500$ and boundary conditions are also identical to the train data.

The simulation domain for the ***out-dis-test* data** was identical with the *in-dis-test* data, except for the boundary condition that was defined as $u(0, t) = 0.7$.

Model Architectures **TCN** is designed to have two input neurons, one hidden layer of size 32, and two output neurons. **ConvLSTM** has two input- and output neurons and one hidden layer with 24 neurons. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and two, respectively, while a hidden layer of size 16 is used. The pure ML models were trained on the first 400 time steps and validated on the remaining 100 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done

Table 6: Closed loop rMSE on the *train* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	5.8×10^{-1}	7.6×10^{-4}	3.4×10^{-4}	3.5×10^{-3}	4.9×10^{-4}	1.8×10^{-3}
2	3.3×10^{0}	3.6×10^{-1}	2.5×10^{-4}	3.7×10^{-5}	4.1×10^{-4}	2.3×10^{-4}
3	1.3×10^{0}	7.1×10^{-1}	4.3×10^{-4}	1.9×10^{-5}	5.2×10^{-4}	1.8×10^{-3}
4	3.2×10^{0}	1.6×10^{-1}	1.5×10^{-3}	3.3×10^{-3}	4.6×10^{-4}	6.6×10^{-4}
5	2.8×10^{-2}	1.4×10^{0}	1.0×10^{-3}	4.0×10^{-5}	1.2×10^{-3}	8.4×10^{-5}
6	4.3×10^{-3}	9.4×10^{-1}	6.4×10^{-4}	2.8×10^{-5}	7.1×10^{-4}	3.1×10^{-4}
7	5.1×10^{-3}	1.0×10^{0}	1.2×10^{-3}	1.4×10^{-4}	2.9×10^{-4}	2.4×10^{-4}
8	2.8×10^{-1}	2.2×10^{-3}	4.1×10^{-4}	3.3×10^{-5}	3.8×10^{-4}	1.9×10^{-3}
9	6.9×10^{0}	3.5×10^{-2}	9.5×10^{-4}	1.4×10^{-4}	3.3×10^{-4}	1.9×10^{-4}
10	1.1×10^{-2}	4.4×10^{-1}	6.2×10^{-4}	1.7×10^{-4}	8.4×10^{-4}	2.7×10^{-4}

Table 7: Closed loop rMSE on *in-dis-test* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	2.9×10^{-1}	1.1×10^{-1}	1.0×10^{-2}	4.4×10^{-1}	5.5×10^{-2}	4.7×10^{-3}
2	3.5×10^{0}	1.5×10^{-1}	1.3×10^{-3}	1.3×10^{-2}	1.2×10^{-1}	5.2×10^{-4}
3	7.0×10^{-1}	3.1×10^{-1}	2.5×10^{-2}	1.4×10^{-3}	4.3×10^{-3}	4.7×10^{-3}
4	3.5×10^{0}	4.2×10^{-1}	5.2×10^{-2}	7.7×10^{-2}	5.3×10^{-3}	1.6×10^{-3}
5	5.2×10^{-1}	1.2×10^{0}	1.1×10^{-1}	1.2×10^{-4}	4.2×10^{-1}	1.4×10^{-4}
6	3.5×10^{-1}	8.0×10^{-1}	4.5×10^{-3}	2.1×10^{-4}	7.1×10^{-1}	7.2×10^{-4}
7	2.0×10^{-2}	6.2×10^{-1}	9.6×10^{-2}	2.7×10^{-3}	1.5×10^{-2}	5.5×10^{-4}
8	5.7×10^{-1}	3.2×10^{-3}	1.4×10^{-2}	1.0×10^{-2}	3.6×10^{-3}	5.0×10^{-3}
9	8.3×10^{0}	2.0×10^{-1}	2.7×10^{-2}	6.6×10^{-2}	3.6×10^{-3}	4.1×10^{-4}
10	5.0×10^{-1}	6.2×10^{-1}	1.5×10^{-2}	5.1×10^{-3}	4.1×10^{-3}	6.1×10^{-4}

Table 8: Closed loop rMSE on *out-dis-test* data from ten different training runs for each model for the diffusion-sorption equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.4×10^{0}	6.8×10^{-1}	8.7×10^{-2}	-	3.9×10^{-1}	2.9×10^{-3}
2	6.4×10^{0}	1.3×10^{0}	4.8×10^{-2}	-	3.2×10^{-1}	4.4×10^{-4}
3	2.8×10^{0}	1.6×10^{0}	1.0×10^{-1}	-	4.1×10^{-1}	2.9×10^{-3}
4	6.0×10^{0}	5.8×10^{-1}	6.0×10^{-2}	-	3.3×10^{-1}	1.0×10^{-3}
5	1.3×10^{0}	4.0×10^{0}	1.4×10^{-2}	-	6.4×10^{-1}	1.5×10^{-4}
6	1.2×10^{-2}	3.0×10^{0}	2.6×10^{-1}	-	1.3×10^{0}	5.6×10^{-4}
7	2.8×10^{-2}	2.8×10^{0}	4.9×10^{-1}	-	4.5×10^{-1}	4.5×10^{-4}
8	1.5×10^{-1}	7.9×10^{-2}	2.9×10^{-2}	-	3.7×10^{-1}	3.1×10^{-3}
9	1.4×10^{1}	1.0×10^{0}	2.2×10^{-1}	-	4.0×10^{-1}	3.7×10^{-4}
10	1.2×10^{0}	2.4×10^{0}	7.8×10^{-2}	-	3.9×10^{-1}	4.9×10^{-4}

Composing Partial Differential Equations with Physics-Aware Neural Networks

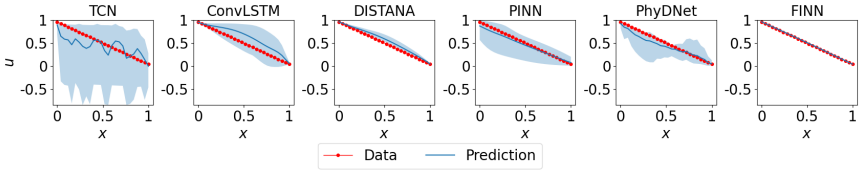


Figure 10: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration in the diffusion-sorption equation at $t = 10\,000$ for the *in-dis-test* dataset.

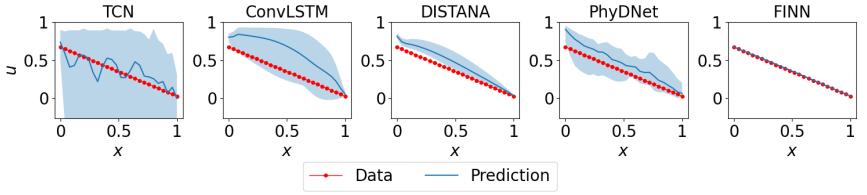


Figure 11: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration in the diffusion-sorption equation at $t = 10\,000$ for the *out-dis-test* dataset.

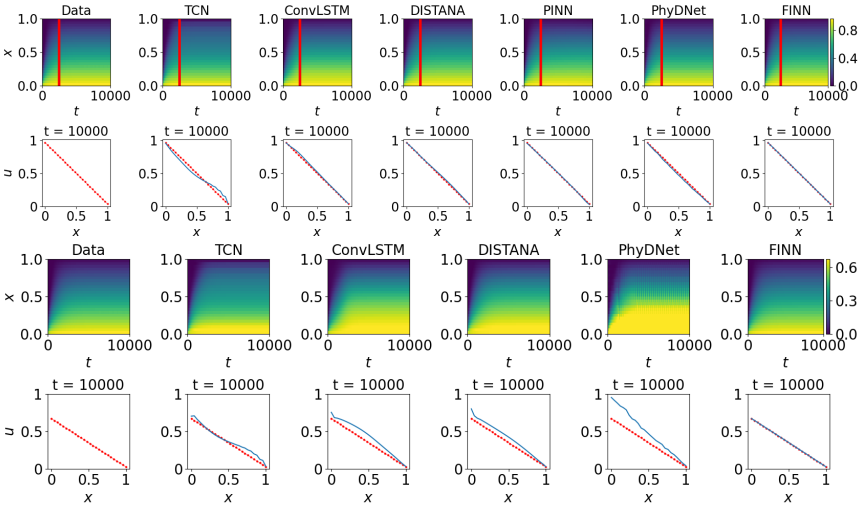


Figure 12: Plots of diffusion-sorption's dissolved concentration (red) and prediction (blue) using different models. The first and third rows show the solution over x and t (vertical red lines mark the transition from *train* to *in-dis-test*). Second (*in-dis-test*) and fourth (*out-dis-test*) rows visualize the solution distributed in x at $t = 10\,000$.

during teacher forcing. **PINN** was defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 2]. **PhyDNet** was defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_{\mathcal{N}}$ and $\varphi_{\mathcal{D}}$ were used, with $\varphi_{\mathcal{D}}$ defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes u as an input and outputs the retardation factor $R(u)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 6), *in-dis-test* (Table 7 and Figure 10), and *out-dis-test* (Table 8 and Figure 11) datasets. Results for the total concentration u_t were omitted due to high similarity to the concentration of the reported contamination solution u.

C.3. Diffusion-Reaction

The diffusion-reaction equation is applicable in physical and biological systems, for example in pattern formation (Turing, 1952).

Data In the current paper, we consider the two-dimensional diffusion-reaction for that class of problems:

$$\frac{\partial u_1}{\partial t} = R_1(u_1, u_2) + D_1 \left(\frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_1}{\partial y^2} \right), \quad (19)$$

$$\frac{\partial u_2}{\partial t} = R_2(u_1, u_2) + D_2 \left(\frac{\partial^2 u_2}{\partial x^2} + \frac{\partial^2 u_2}{\partial y^2} \right). \quad (20)$$

Here, $D_1 = 10^{-3}$ and $D_2 = 5 \times 10^{-3}$ are the diffusion coefficient for the activator and inhibitor, respectively. The system of equations is coupled through the reaction terms $R_1(u_1, u_2)$ and $R_2(u_1, u_2)$ which are both dependent on u_1 and u_2. In this work, the Fitzhugh–Nagumo (Klaasen & Troy, 1984) system was considered to define the reaction function:

$$R_1(u_1, u_2) = u_1 - u_1^3 - k - u_2, \quad (21)$$

$$R_2(u_1, u_2) = u_1 - u_2, \quad (22)$$

with $k = 5 \times 10^{-3}$.

The simulation domain for the ***train* data** is defined with $x = [-1, 1]$, $y = [-1, 1]$, $t = [0, 10]$ and is discretized with $N_x = 49$ and $N_y = 49$ spatial locations, and $N_t = 101$ simulation steps. The initial condition was defined as $u_1(x, 0) = u_2(x, 0) = \sin(\pi(x + 1)/2) \sin(\pi(y + 1)/2)$, and the corresponding boundary condition was defined as $\frac{\partial u_1}{\partial x}(-1, y, t) = 0$, $\frac{\partial u_1}{\partial x}(1, y, t) = 0$, $\frac{\partial u_2}{\partial x}(-1, y, t) = 0$, $\frac{\partial u_2}{\partial x}(1, y, t) = 0$, $\frac{\partial u_1}{\partial y}(x, -1, t) = 0$, $\frac{\partial u_1}{\partial y}(x, 1, t) = 0$, $\frac{\partial u_2}{\partial y}(x, -1, t) = 0$, and $\frac{\partial u_2}{\partial y}(x, 1, t) = 0$.

The ***in-dis-test* data** is simulated with with $x = [-1, 1]$, $y = [-1, 1]$ and with the time span of $t = [10, 50]$ and $N_t = 401$. Initial condition is taken from the train data at $t = 10$ and boundary conditions are also identical to the train data.

The simulation domain for the ***out-dis-test* data** was identical with the train data, except for the initial condition that was defined as $u_1(x, 0) = u_2(x, 0) = \sin(\pi(x + 1)/2) \sin(\pi(y + 1)/2) - 0.5$, i.e. subtracting 0.5 from the original initial condition.

Model Architectures **TCN** is designed to have two input- and output neurons, and one hidden layer of size 32. **ConvLSTM** has two input- and output neurons and one hidden layer of size 24. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and two, respectively, while a hidden layer of size 16 is used. The pure ML models were trained on the first 70 time steps and validated on the remaining 30 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done during teacher forcing. **PINN** is defined as a feedforward network with the size of [3, 20, 20, 20, 20, 20, 20, 20, 20, 2]. **PhyDNet** is defined with the PhyCell containing 32 input dimensions, 49 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer. For **FINN**, the modules $\varphi_{\mathcal{N}}$, $\varphi_{\mathcal{D}}$ and Φ are used, with $\varphi_{\mathcal{D}}$ set as two learnable scalars that learn the diffusion coefficients D_1 and D_2, and Φ defined as a feedforward network with the size of [2, 20, 20, 20, 2] that takes u_1 and u_2 as inputs and outputs the reaction functions $R_1(u_1, u_2)$ and $R_2(u_1, u_2)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 9), *in-dis-test* (Table 10 and Figure 13), and *out-dis-test* (Table 11 and Figure 14) datasets. Results for the total inhibitor u_2 were omitted due to high similarity to the reported activator u_1.

C.4. Allen–Cahn

The Allen–Cahn equation is commonly employed in reaction-diffusion systems, e.g. to model phase separation in multi-component alloy systems (Raissi et al., 2019).

Table 9: Closed loop rMSE on *train* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.2×10^{-1}	2.3×10^{-2}	2.3×10^{-2}	3.7×10^{-3}	1.1×10^{-3}	1.3×10^{-3}
2	5.9×10^{-2}	2.2×10^{-2}	5.6×10^{-2}	7.4×10^{-3}	9.5×10^{-4}	2.1×10^{-3}
3	4.3×10^{-1}	1.9×10^{-2}	1.2×10^{-1}	2.8×10^{-3}	7.7×10^{-4}	1.8×10^{-3}
4	1.2×10^{-1}	1.0×10^{-2}	8.7×10^{-3}	3.7×10^{-3}	9.1×10^{-4}	1.6×10^{-3}
5	3.3×10^{-1}	1.9×10^{-2}	6.2×10^{-3}	3.8×10^{-3}	1.0×10^{-3}	1.4×10^{-3}
6	1.1×10^{-1}	6.4×10^{-3}	2.9×10^{-3}	3.3×10^{-3}	8.7×10^{-4}	2.3×10^{-3}
7	3.2×10^{-1}	5.0×10^{-2}	1.1×10^{-1}	68.1×10^{-4}	1.1×10^{-3}	2.2×10^{-3}
8	1.5×10^{-1}	1.4×10^{-2}	3.2×10^{-2}	1.3×10^{-3}	1.0×10^{-3}	1.3×10^{-3}
9	1.8×10^{-1}	2.9×10^{-2}	5.0×10^{-2}	4.2×10^{-4}	1.2×10^{-3}	2.0×10^{-3}
10	7.4×10^{-2}	9.7×10^{-1}	1.2×10^{-1}	8.3×10^{-3}	9.9×10^{-4}	1.5×10^{-3}

Table 10: Closed loop rMSE on *in-dis-test* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.4×10^{0}	6.4×10^{-1}	1.3×10^{0}	6.1×10^{-1}	4.9×10^{-1}	2.3×10^{-2}
2	4.7×10^{0}	4.6×10^{-1}	1.5×10^{0}	1.8×10^{0}	7.0×10^{-1}	1.3×10^{-2}
3	5.2×10^{0}	7.4×10^{-1}	2.4×10^{0}	2.8×10^{-1}	6.2×10^{-1}	1.4×10^{-2}
4	2.4×10^{0}	5.3×10^{-1}	1.6×10^{0}	2.1×10^{-1}	5.4×10^{-1}	2.3×10^{-2}
5	5.8×10^{0}	3.9×10^{-1}	1.5×10^{0}	1.0×10^{0}	4.2×10^{-1}	1.3×10^{-2}
6	1.5×10^{0}	4.3×10^{-1}	2.2×10^{-1}	7.1×10^{-1}	7.9×10^{-1}	1.6×10^{-2}
7	4.8×10^{0}	7.9×10^{-1}	1.7×10^{0}	4.3×10^{-1}	5.3×10^{-1}	1.2×10^{-2}
8	5.7×10^{0}	6.9×10^{-1}	1.5×10^{0}	1.0×10^{-1}	7.3×10^{-1}	1.9×10^{-2}
9	4.4×10^{0}	1.8×10^{0}	1.3×10^{0}	1.8×10^{-1}	5.3×10^{-1}	1.5×10^{-2}
10	1.7×10^{0}	9.4×10^{-1}	1.3×10^{0}	2.1×10^{-1}	8.9×10^{-1}	1.7×10^{-2}

Table 11: Closed loop rMSE on *out-dis-test* data from ten different training runs for each model for the diffusion-reaction equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	6.7×10^{-1}	9.2×10^{-2}	2.7×10^{-1}	-	1.1×10^{0}	1.9×10^{-1}
2	4.9×10^{0}	3.7×10^{-1}	3.9×10^{-1}	-	1.6×10^{0}	1.6×10^{-1}
3	7.4×10^{0}	3.8×10^{-1}	6.1×10^{-1}	-	7.2×10^{-1}	1.7×10^{-1}
4	2.1×10^{0}	2.1×10^{-1}	2.3×10^{-1}	-	1.2×10^{0}	1.9×10^{-1}
5	7.4×10^{0}	1.7×10^{-1}	7.5×10^{-2}	-	8.0×10^{-1}	1.8×10^{-1}
6	1.9×10^{0}	4.4×10^{-1}	6.0×10^{-2}	-	8.3×10^{-1}	1.8×10^{-1}
7	6.5×10^{0}	3.0×10^{-1}	5.6×10^{-1}	-	8.3×10^{-1}	1.7×10^{-1}
8	5.7×10^{0}	1.1×10^{-1}	3.4×10^{-1}	-	6.2×10^{-1}	1.8×10^{-1}
9	6.6×10^{0}	1.2×10^{0}	3.7×10^{-1}	-	6.0×10^{-1}	1.7×10^{-1}
10	1.1×10^{0}	1.1×10^{0}	9.7×10^{-1}	-	1.7×10^{0}	1.8×10^{-1}

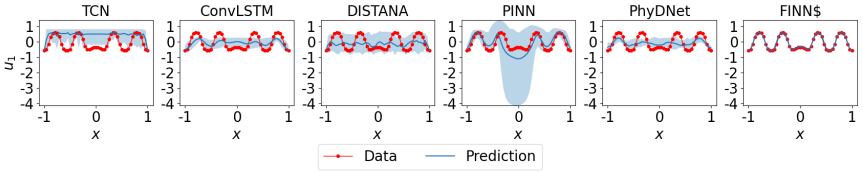


Figure 13: Prediction mean over ten different trained models (with 95% confidence interval) of the activator in the diffusion-reaction equation at $t = 50$ for the *in-dis-test* dataset.

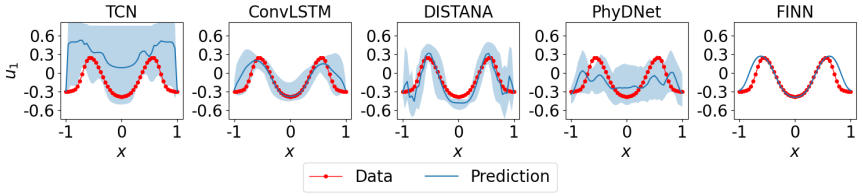


Figure 14: Prediction mean over ten different trained models (with 95% confidence interval) of the activator in the diffusion-reaction equation at $t = 10$ for the *out-dis-test* dataset.

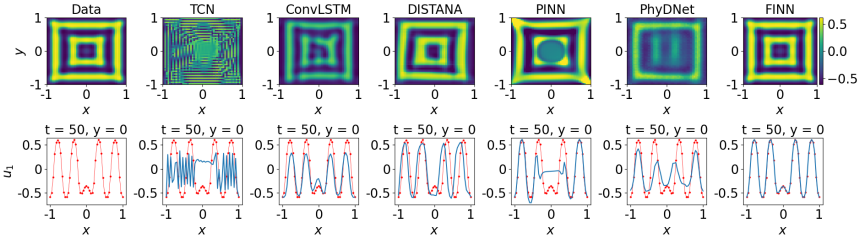


Figure 15: Plots of diffusion-reaction's activator data u_1 (red) and extrapolated prediction (blue) using different models. The plots in the first row show the solution distributed over x and y at $t = 50$, and the plots in the second row show the solution distributed in x at $y = 0$ and $t = 50$.

Composing Partial Differential Equations with Physics-Aware Neural Networks

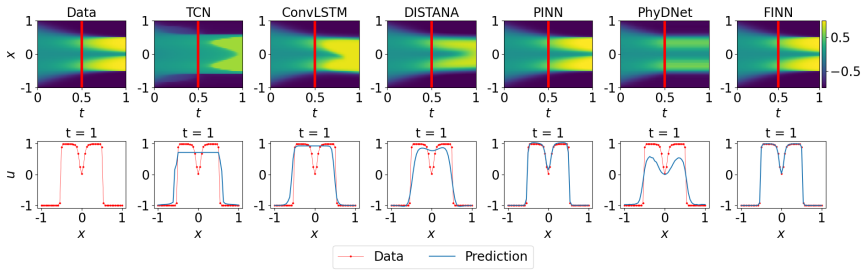


Figure 16: Plots of Allen–Cahn's data and prediction of *in-dis-test* data using different models. The plots in the first row show the solution over x and t (the red lines mark the transition from *train* to *in-dis-test*), the second row visualizes the best model's solution distributed in x at $t = 1$.

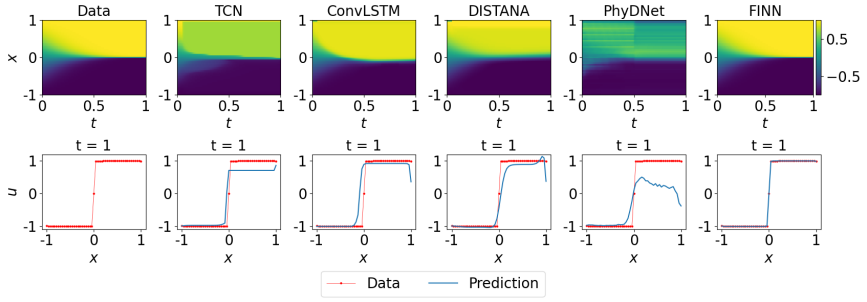


Figure 17: Plots of Allen–Cahn's data and prediction of *out-dis-test* data using different models. The plots in the first row show the solution over x and t, the second row visualizes the solution distributed in x at $t = 1$.

Data The one-dimensional Allen–Cahn equation is written as

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + R(u), \tag{23}$$

where the main unknown is u, the reaction term is denoted as $R(u)$ which is a function of u and the diffusion coefficient is $D = 10^{-4}$. In the current work, the reaction term is defined as:

$$R(u) = 5u - 5u^3, \tag{24}$$

to reproduce the experiment conducted in the PINN paper (Raissi et al., 2019). The simulation domain for the ***train* data** is defined with $x = [-1, 1]$, $t = [0, 0.5]$ and is discretized with $N_x = 49$ spatial locations, and $N_t = 201$ simulation steps. The initial condition is defined as $u(x, 0) = x^2\cos(\pi x)$, and periodic boundary condition is used, i.e. $u(-1, t) = u(1, t)$.

***In-dis-test* data** is simulated with $x = [-1, 1]$, a time span of $t = [0.5, 1]$ and $N_t = 201$. Initial condition is taken from the train data at $t = 0.5$ and boundary conditions are also similar to the train data.

The simulation domain for the ***out-dis-test* data** is identical with the train data, except for the initial condition that is defined as $u(x, 0) = \sin(\pi x/2)$.

Model Architectures The **TCN** and **ConvLSTM** are designed to have one input neuron, one hidden layer of size 32 and 24, respectively, and one output neuron. The lateral and dynamic input- and output sizes of the **DISTANA** model are set to one and a hidden layer of size 16 is used. The pure ML models were trained on the first 150 time steps and validated on the remaining 50 time steps of the train data (applying early stopping). Also, to prevent the pure ML models from diverging too much in closed loop, the boundary data are fed into the models as done dur-

Table 12: Closed loop rMSE on the *train* data from ten different training runs for each model for the Allen–Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	5.0×10^{-2}	3.2×10^{-1}	1.3×10^{-3}	7.5×10^{-5}	6.6×10^{-4}	8.0×10^{-5}
2	2.0×10^{-1}	1.4×10^{-1}	2.1×10^{-3}	1.9×10^{-4}	2.9×10^{-4}	3.7×10^{-5}
3	1.3×10^{-1}	5.5×10^{-2}	3.6×10^{-3}	1.9×10^{-4}	1.1×10^{-3}	2.5×10^{-5}
4	4.0×10^{-1}	3.0×10^{-2}	2.2×10^{-3}	7.1×10^{-6}	2.2×10^{-4}	2.1×10^{-6}
5	6.2×10^{-2}	3.6×10^{-1}	1.2×10^{-3}	6.0×10^{-5}	2.3×10^{-4}	1.1×10^{-4}
6	4.8×10^{-2}	2.1×10^{-2}	2.6×10^{-3}	2.9×10^{-5}	2.2×10^{-4}	1.1×10^{-5}
7	4.4×10^{-2}	3.2×10^{-2}	1.9×10^{-3}	6.4×10^{-5}	7.9×10^{-4}	5.2×10^{-5}
8	1.7×10^{-1}	2.4×10^{-3}	3.9×10^{-3}	9.5×10^{-5}	1.7×10^{-4}	1.3×10^{-6}
9	3.1×10^{0}	1.8×10^{0}	3.5×10^{-3}	9.0×10^{-5}	4.5×10^{-4}	1.4×10^{-6}
10	1.3×10^{-1}	8.1×10^{-4}	6.8×10^{-4}	1.7×10^{-4}	4.5×10^{-4}	1.3×10^{-6}

Table 13: Closed loop rMSE on *in-dis-test* data from ten different training runs for each model for the Allen–Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	1.5×10^{-1}	1.7×10^{0}	7.0×10^{-2}	1.0×10^{-2}	2.0×10^{-1}	8.2×10^{-5}
2	4.2×10^{-1}	1.3×10^{0}	1.3×10^{-1}	2.7×10^{-1}	1.2×10^{-1}	3.6×10^{-5}
3	3.4×10^{-1}	5.3×10^{-1}	1.2×10^{-1}	6.2×10^{-1}	1.5×10^{-1}	2.3×10^{-5}
4	6.8×10^{-1}	4.9×10^{-1}	1.1×10^{-1}	6.7×10^{-3}	1.4×10^{-1}	3.0×10^{-6}
5	2.8×10^{-1}	8.0×10^{-1}	1.1×10^{-1}	3.3×10^{-3}	1.2×10^{-1}	1.4×10^{-4}
6	1.8×10^{-1}	4.6×10^{-1}	1.5×10^{-1}	3.0×10^{-3}	1.8×10^{-1}	1.6×10^{-5}
7	1.8×10^{-1}	7.8×10^{-1}	1.8×10^{-1}	7.3×10^{-4}	3.3×10^{-1}	4.9×10^{-5}
8	4.4×10^{-1}	8.2×10^{-1}	1.4×10^{-1}	1.9×10^{-1}	1.5×10^{-1}	1.9×10^{-6}
9	2.1×10^{0}	1.9×10^{0}	2.4×10^{-1}	1.8×10^{-2}	1.5×10^{-1}	1.9×10^{-6}
10	3.4×10^{-1}	8.0×10^{-2}	6.8×10^{-2}	4.8×10^{-2}	1.4×10^{-1}	1.9×10^{-6}

Table 14: Closed loop rMSE on *out-dis-test* data from ten different training runs for each model for the Allen–Cahn equation.

Run	TCN	ConvLSTM	DISTANA	PINN	PhyDNet	FINN
1	4.8×10^{-2}	6.1×10^{-2}	1.6×10^{-2}	-	7.2×10^{-1}	8.6×10^{-5}
2	2.3×10^{-1}	4.4×10^{-1}	4.3×10^{-2}	-	2.8×10^{-1}	3.4×10^{-5}
3	1.5×10^{-1}	1.7×10^{-1}	7.3×10^{-2}	-	7.3×10^{-1}	2.1×10^{-5}
4	2.3×10^{-1}	3.4×10^{-1}	1.2×10^{-2}	-	8.5×10^{-1}	3.5×10^{-6}
5	2.0×10^{-1}	9.7×10^{-2}	4.7×10^{-2}	-	7.9×10^{-1}	1.5×10^{-4}
6	1.0×10^{-1}	3.4×10^{-1}	3.6×10^{-2}	-	9.1×10^{-1}	1.7×10^{-5}
7	1.2×10^{-1}	1.8×10^{-1}	6.9×10^{-2}	-	1.2×10^{0}	5.0×10^{-5}
8	1.7×10^{-1}	6.2×10^{-1}	1.1×10^{-1}	-	6.3×10^{-1}	2.3×10^{-6}
9	1.4×10^{0}	1.6×10^{0}	1.2×10^{-1}	-	7.2×10^{-1}	2.1×10^{-6}
10	1.1×10^{-1}	8.8×10^{-2}	1.8×10^{-2}	-	6.9×10^{-1}	2.2×10^{-6}

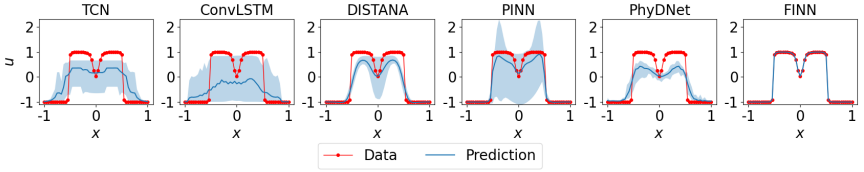


Figure 18: Prediction mean over ten different trained models (with 95% confidence interval) of the Allen–Cahn equation at $t = 1$ for the *in-dis-test* dataset.

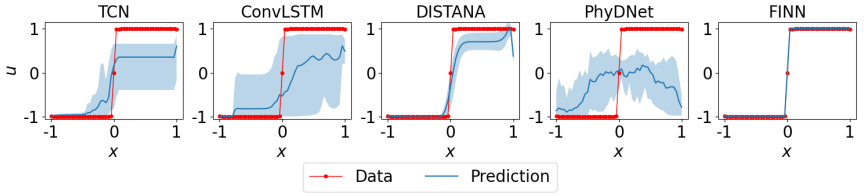


Figure 19: Prediction mean over ten different trained models (with 95% confidence interval) of the Allen–Cahn equation at $t = 1$ for the *out-dis-test* data.

ing teacher forcing. **PINN** was defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 1] (8 hidden layers, each contains 20 hidden neurons). **PhyDNet** was defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer.

For **FINN**, the modules $\varphi_{\mathcal{N}}$, $\varphi_{\mathcal{D}}$, and Φ were used, with $\varphi_{\mathcal{D}}$ defined as a learnable scalar that learns the diffusion coefficient D, and Φ defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes u as an input and outputs the reaction function $R(u)$. All models are trained until convergence using the L-BFGS optimizer, except for PhyDNet, which is trained with the Adam optimizer and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer.

Additional Results Individual errors are reported for the ten different training runs and visualizations are generated for the *train* (Table 12), *in-dis-test* (Table 13, Figure 16 and Figure 18), and *out-dis-test* (Table 14, Figure 17 and Figure 19) datasets.

C.5. Soil Parameters and Simulation Domains for the Experimental Dataset

Identical to Praditia et al. (2021), the soil parameters and simulation (and experimental) domain used in the real-world diffusion-sorption experiment are summarized in Table 15 for core samples #1, #2, and #2B.

Table 15: Soil and experimental parameters of core samples #1, #2, and #2B. D is the diffusion coefficient, ϕ is the porosity, ρ_s is the bulk density, L and r are the length and radius of the sample, t_{end} is the simulation time, Q is the flow rate in the bottom reservoir and u_s is the total concentration of trichloroethylene in the sample.

Soil parameters				
Param.	Unit	Core #1	Core #2	Core #2B
D	m^2/day	$2.00 \cdot 10^{-5}$	$2.00 \cdot 10^{-5}$	$2.78 \cdot 10^{-5}$
ϕ	-	0.288	0.288	0.288
ρ_s	kg/m^3	1957	1957	1957
Simulation domain				
Param.	Unit	Core #1	Core #2	Core #2B
L	m	0.0254	0.02604	0.105
r	m	0.02375	0.02375	N/A
t_{end}	days	38.81	39.82	48.88
Q	m^3/day	$1.01 \cdot 10^{-4}$	$1.04 \cdot 10^{-4}$	N/A
u_s	kg/m^3	1.4	1.6	1.4

For all experiments, the core samples are subjected to a constant contaminant concentration at the top u_s, which can be treated as a Dirichlet boundary condition numerically. Notice that, for core sample #2, we set u_s to be

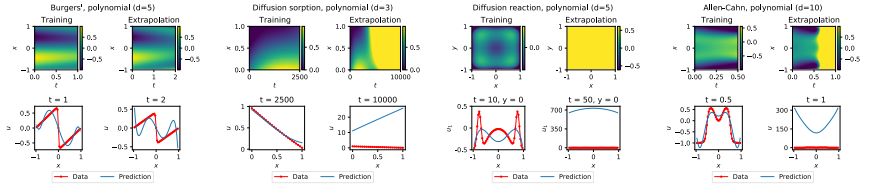


Figure 20: Prediction pairs for *train* and *in-dis-test* (left and right columns of the pairs) of the Burgers' (left, polynomial order 5), diffusion-sorption (second left, polynomial order 3), diffusion-reaction (second right, polynomial order 5 since higher orders did not converge), and Allen–Cahn (right, polynomial order 10) equations using polynomial regression.

slightly higher to compensate for the fact that there might be fractures at the top of core sample #2, so that the contaminant can break through the core sample faster.

For core samples #1 and #2, Q is the flow rate of clean water at the bottom reservoir that determines the Cauchy boundary condition at the bottom of the core samples. For core sample #2B, note that the sample length is significantly longer than the other samples, and by the end of the experiment, no contaminant has broken through the core sample. Therefore, we assume the bottom boundary condition to be a no-flow Neumann boundary condition; see (Praditia et al., 2021) for details.

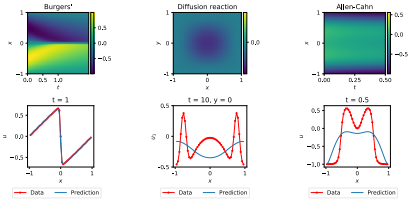
D. Ablations

We conduct the ablations with a one-dimensional instead of the two-dimensional domain for Burgers' equation. The diffusion coefficient is $D = 0.01/\pi$, and the simulation domain for the ***train* data** is defined with $x = [-1, 1]$, $t = [0, 1]$ and is discretized with $N_x = 49$ spatial locations, and $N_t = 201$ simulation steps. The initial condition is defined as $u(x, 0) = -\sin(\pi x)$, and the boundary condition is defined as $u(-1, t) = u(1, t) = 0$. ***In-dis-test* data** is simulated with $x = [-1, 1]$ and a time span of $t = [1, 2]$ and $N_t = 201$. Initial condition is taken from the *train* data at $t = 1$ and boundary conditions are also similar to the *train* data. The simulation domain for the ***out-dis-test* data** is identical with the *train* data, except for the initial condition that is defined as $u(x, 0) = \sin(\pi x)$.

D.1. Baseline Assessment with Polynomial Regression

In this section, we apply polynomial regression to show that the example problems chosen in this work (i.e. Burgers', diffusion-sorption, diffusion-reaction, and Allen–Cahn) are not easy to solve. First, we use polynomial regression to fit the unknown variable $u = f(x, t)$, similar to PINN. Figure 20 shows the prediction of u for each example, obtained using the fitted polynomial coefficients. For the Burgers' equation, the *train* and *in-dis-test* predictions have rMSE values of 1.4×10^{-1} and 3.3×10^{-1}, respectively. For the diffusion-sorption equation, the *train* and *in-dis-test* predictions have rMSE values of 3.4×10^{-2} and 4.1×10^{1}, respectively. For the diffusion-reaction equation, the *train* and *in-dis-test* predictions have rMSE values of 2.2×10^{-1} and 7.1×10^{2}, respectively. For the Allen–Cahn equation, the *train* and *in-dis-test* predictions have rMSE values of 4.0×10^{-2} and 2.6×10^{5}, respectively. The simple polynomial fitting fails to obtain accurate predictions of the solution for all example problems. The results also show that the polynomials overfit the data, evidenced by the significant deterioration of performance during extrapolation (prediction of *in-dis-test* data). The diffusion-reaction and the Allen–Cahn equations are particularly the most difficult to fit, because they require higher order polynomials to obtain reasonable accuracy. With the high order, they still fail to even fit the *train* data well.

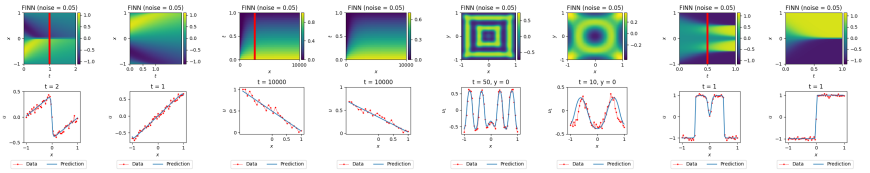
Next, we also consider using polynomial fitting in lieu of ANNs (namely the modules $\varphi_{\mathcal{A}}$, $\varphi_{\mathcal{D}}$, and Φ) in FINN. In-



(a) (b) (c)

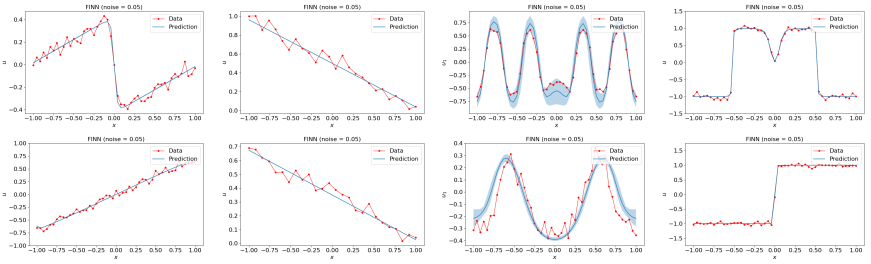
Figure 21: Plots of the *train* prediction in the Burgers' (left), diffusion-reaction (center), and Allen–Cahn equations (right) using FINN with polynomial fitting. Due to instability issues, the diffusion-sorption equation could not be solved with the polynomial FINN. The plots in the first row show the solution over x and t, and the plots in the second row show the solution distributed in x.

Composing Partial Differential Equations with Physics-Aware Neural Networks



(a) Burgers' (b) Diffusion-sorption (c) Diffusion-reaction (d) Allen–Cahn

Figure 22: Paired plots of the *in-dis-test* and *out-dis-test* prediction (left and right of the pairs, respectively) after training FINN with noisy data. The plots in the first row of the pairs show the solution over x and t (red line marks the transition from *train* to *in-dis-test*), and the plots in the second row of the pairs show the best model's solution distributed in x.



(a) Burgers' (b) Diffusion-sorption (c) Diffusion-reaction (d) Allen–Cahn

Figure 23: Prediction mean over ten different trained FINN (with 95% confidence interval) of the (from left to right) one-dimensional Burgers', diffusion-sorption, diffusion-reaction, and Allen–Cahn equations obtained by training FINN with noisy data for the *in-dis-test* and *out-dis-test* (top and bottom, accordingly) prediction.

deed, with this method, the model successfully predicts Burgers' equation (Figure 21a). The rMSE values are 5.4×10^{-4}, 1.9×10^{-2}, and 3.6×10^{-4} for *train*, *in-dis-test*, and *out-dis-test* data, respectively. However, the model fails to complete the training for the diffusion-sorption equation due to major instabilities (the polynomials can produce negative outputs, leading to negative diffusion coefficients that cause numerical instability). Moreover, the model also fails to sufficiently learn the diffusion-reaction (Figure 21b) and the Allen–Cahn (Figure 21c) equations. For the diffusion-reaction equation, the rMSE values are 3.3×10^{-1}, 1.6×10^{0}, and 1.4×10^{0} for *train*, *in-dis-test*, and *out-dis-test* data, respectively. For the Allen–Cahn equation, the rMSE values are 2.6×10^{-1}, 6.6×10^{-1}, and 5.0×10^{-1} for *train*, *in-dis-test*, and *out-dis-test* data, respectively. Even though the unknown equations do not seem too complicated, they are still difficult to solve together with the PDE as a whole. The results show that for these particular problems, ANNs serve better because they allow better control during training in form of constraints, and they produce more regularized outputs than high order polynomials. However, while ANNs are not unique in their selection, they are most convenient for our implementation.

D.2. Noise Robustness Test of FINN

In this section, we test the robustness of FINN when trained using noisy data. All the synthetic data is generated with the same parameters, only added with noise from the distribution $\mathcal{N}(0.0, 0.05)$. For the Burgers' equation (Figure 22a and Figure 23a), the average rMSE values are $6.9\times10^{-3}\pm1.1\times10^{-5}$, $2.5\times10^{-2}\pm6.6\times10^{-5}$, and $7.1\times10^{-3}\pm1.1\times10^{-5}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the diffusion-sorption equation (Figure 22b and Figure 23b), the average rMSE values are $3.9\times10^{-2}\pm6.9\times10^{-5}$, $3.6\times10^{-2}\pm5.3\times10^{-5}$, and $7.1\times10^{-2}\pm1.0\times10^{-4}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the diffusion-reaction equation (Figure 22c and Figure 23c), the average rMSE values are $4.1\times10^{-2}\pm6.8\times10^{-3}$, $1.3\times10^{-1}\pm6.9\times10^{-2}$, and $2.2\times10^{-1}\pm1.3\times10^{-2}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. For the Allen–Cahn

Composing Partial Differential Equations with Physics-Aware Neural Networks

Table 16: Comparison of rMSE on the *Train* data for individual runs with and without Neural ODE (NODE) on the different equations. Average of models without (w/o) NODE is calculated over all individual runs. Average scores of models with (w/) NODE are taken from Table 1. Left and right columns for the different equations show the MSE and epoch from which on `NaN` values occured, respectively. FINN was trained for 100 epochs, overall, i.e. 100 in the second columns corresponds to a training without any `NaN` values encountered.

Run	Burgers' (1D)		Diffusion-sorption		Diffusion-reaction		Allen–Cahn	
	rMSE	Ep.	rMSE	Ep.	rMSE	Ep.	rMSE	Ep.
1	1.7×10^{-5}	100	-	0	3.3×10^{-1}	2	3.0×10^{-5}	100
2	2.2×10^{-5}	100	-	0	1.7×10^{-2}	14	2.8×10^{-4}	13
3	2.8×10^{-3}	100	-	0	8.4×10^{-3}	27	5.0×10^{-5}	100
4	4.0×10^{-5}	100	-	0	5.2×10^{-2}	78	2.1×10^{-6}	100
5	3.7×10^{-3}	2	-	0	5.7×10^{-2}	100	3.8×10^{-4}	100
6	1.7×10^{-5}	100	-	0	2.4×10^{-1}	83	1.2×10^{-6}	100
7	1.3×10^{-5}	100	-	0	1.1×10^{-2}	31	7.3×10^{-7}	100
8	3.4×10^{-3}	3	-	0	1.3×10^{-2}	15	2.5×10^{-6}	100
9	4.0×10^{-5}	13	-	0	3.1×10^{-3}	100	2.5×10^{-6}	100
10	1.4×10^{-5}	100	-	0	2.4×10^{-3}	45	3.3×10^{-5}	39
Avg w/o NODE	$(1.0 \pm 1.5) \times 10^{-3}$	72	-	0	$(7.4 \pm 11.0) \times 10^{-2}$	50	$(7.8 \pm 12.9) \times 10^{-5}$	85
Avg w/ NODE	$(7.8 \pm 8.2) \times 10^{-6}$	100	(omitted)	100	$(1.7 \pm 0.4) \times 10^{-3}$	100	$(3.2 \pm 3.5) \times 10^{-5}$	100

equation (Figure 22d and Figure 23d), the average rMSE values are $1.2 \times 10^{-2} \pm 4.8 \times 10^{-6}$, $3.6 \times 10^{-3} \pm 1.3 \times 10^{-5}$, and $2.9 \times 10^{-3} \pm 7.3 \times 10^{-6}$ for the *train*, *in-dis-test*, and *out-dis-test* prediction, respectively. These results show that even though FINN is trained with noisy data, it is still able to capture the essence of the equation and generalize well to different initial and boundary conditions. Additionally, the prediction is consistent, shown by the low values of the rMSE standard deviation, as well as the very narrow confidence interval in the plots.

D.3. Neural ODE Ablation and Runtime Analysis

Here, we investigate whether FINN or Neural ODE (NODE) alone can account for the high accuracy. We find that neither FINN nor NODE alone reach satisfactory results, and it is thus the combination of FINN's modular structure with NODE's adaptive time stepping mechanism that leads to the reported performance.

NODE without FINN The relevance of FINN is demonstrated by an experiment where NODE is applied on top of a conventional CNN stem to take the role of FINN. Results are reported in Table 2 and demonstrate the limitations of a CNN-NODE black box model.

FINN without NODE In order to stress the role of Neural ODE in FINN, we conduct an ablation by removing the Neural ODE part and directly feeding the output of FINN as input in the next time. This amounts to traditional closed loop training or an Euler ODE integration scheme, where the model directly predicts and outputs the delta from $u^{(t)}$ to $u^{(t+1)}$ (also referred to as residual training). Results are summarized in Table 16 and unveil two major effects of Neural ODE.

First, it helps stabilizing the training reasonably: We observe FINN having immense difficulties without Neural ODE at learning the diffusion-sorption equation (not even completing a single epoch without producing `NaN` values). Similar but less dramatic failures were observed on the Burgers', diffusion-reaction, and Allen–Cahn equations, where FINN without Neural ODE on average only managed 72, 50, and 85 epochs, respectively, without producing `NaN` values. This consolidates the supportive aspect of the Neural ODE component, which we explain is due to its adaptive time-stepping feature. More importantly, the adaptive time-stepping of Neural ODE makes it possible to apply FINN to experimental data in the first place, where the time deltas between successive measurements are not constant.

Second and as to be expected, the runtime when adding Neural ODE in the computations increases significantly. We compare the runtime for each model, run on a CPU with i9-9900K core, a clock speed of 3.60 GHz, and 32 GB RAM. Additionally, we also perform the comparison of GPU runtime on a GTX 1060 (i.e. with 6 GB VRAM). The results are summarized in Table 17. Note that the purpose of this comparison is not an optimized benchmark, but only to show that the runtime of FINN is comparable with the other models, especially when run in CPU. When run on GPU, however, FINN runs slightly slower. This is caused by the fact that FINN's implementation is not yet optimized for GPU. More importantly, the Neural ODE package we use benefits only from a larger batch size. As shown in

Composing Partial Differential Equations with Physics-Aware Neural Networks

Table 17: Forward pass runtimes (in seconds) of a single sequence for pure ML and physics-aware neural network methods on the different equations. CPU: Intel i9-9900K, 3.60 GHz processor, GPU: Nvidia GeForce GTX 1060. Note that the datasets (equations) have different sequence lengths. Moreover, the data for PINN on the diffusion-reaction equation did not fit on the GPU and due to instability issues, APHYNITY could not be trained on the diffusion-sorption equation.

		Pure ML models					Physics-aware neural networks			
Equation	Unit	TCN	CNN-NODE	ConvLSTM	DISTANA	FNO	PINN	PhyDNet	APHYNITY	FINN
Burgers' (1D)	CPU	0.45	0.19	0.03	0.04	0.10	0.03	0.11	0.24	0.07
	GPU	0.13	0.33	0.06	0.08	0.20	0.01	0.20	0.45	0.17
Diffusion-sorption (1D)	CPU	5.75	1.34	0.32	0.41	1.03	0.28	1.06	N/A	2.45
	GPU	1.58	2.35	0.58	0.68	2.00	0.01	1.94	N/A	6.10
Diffusion-reaction (2D)	CPU	14.00	1.05	0.18	0.14	0.27	2.47	0.16	0.37	0.31
	GPU	1.01	0.56	0.03	0.04	0.14	N/A	0.12	0.26	0.37
Allen-Cahn (1D)	CPU	1.71	0.10	0.07	0.08	0.21	0.06	0.22	0.45	0.05
	GPU	0.27	0.20	0.13	0.16	0.41	0.01	0.40	0.78	0.14

Figure 24, GPU is faster for batch size larger than 20 000, whereas the maximum size that we use in the example is 2 401. With smaller batch size, CPU usage is faster.

In general, only the PINN model has a benefit when computed on the GPU, since the function is only called once on all batches. This is different for all other models that have to unroll a prediction of the sequence into the future recurrently (except for TCN which is a convolution approach that is faster on GPU). Accordingly, The overhead of copying the tensors to GPU outweighs the GPU's parallelism benefit, compared to directly processing the sequence on the CPU iteratively. On the two-dimensional diffusion-reaction benchmark, the GPU's speed-up comes into play, since in here, the simulation domain is discretized into $49 \times 49 = 2401$ volumes; compared to 49 for the Burgers' (1D) and Allen–Cahn, and 26 for the diffusion-sorption equations. Note that we have observed significantly varying runtimes on different GPUs (i.e. up to three seconds for FINN on Burgers' on an RTX 3090), which might be caused by lack of support of certain packages for a particular hardware, but further investigation is required.

Additionally, we want to emphasize that the higher runtime of FINN on GPU is not caused by the time step adaptivity. In fact, employing the adaptive time stepping strategy is cheaper than choosing all time steps to be small enough (to guarantee numerical stability). As we learn a PDE, we have no exact knowledge to derive a dedicated time integration scheme, but the adaptive Runge–Kutta method is one of the best generic choices. As our PDE and its characteristics change during training, time step adaptivity is a real asset, as, for example, the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1967) would consistently change throughout the training. Therefore, the time step adaptivity is not a bottleneck, but rather a solution for a more efficient computation.

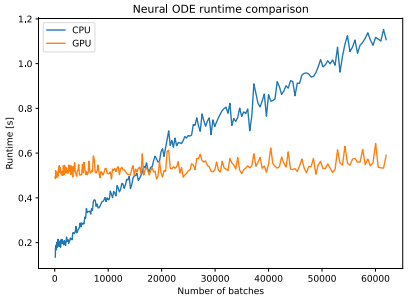


Figure 24: Runtime comparison of Neural ODE with a single hidden layer consisting of 50 hidden nodes run on CPU and GPU for 1 000 time steps. The benefit of a GPU unfolds when larger batch sizes ($> 20\,000$) are used.

In relation to FINN's limitation (see subsection B.2), topics like numerical stabilization schemes for larger time steps, adaptive spatial grid, optimized implementation in a High Performance Computing (HPC) setting, parallelization, etc. hold large potential for future work.

D.4. Training PINN With Finer Spatial Resolution

In order to determine whether the reduced accuracy of PINN in our experiments was caused by a coarse spatial resolution—we only used 49, 26, and 49×49 spatial positions for Burgers' (1D), diffusion-sorption, and diffusion-reaction—another experiment was conducted where the spatial resolutions were increased to $N_x = 999$, $N_x = 251$, and $N_x = 99$, $N_y = 99$, respectively. As reported in Table 18, Figure 25, and Figure 27 (top), the performance

Table 18: rMSE of PINN trained on data with finer resolution from ten different training runs for the diffusion-reaction equation.

Run	*Train*	*In-dis-test*	*Out-dis-test*
1	1.9×10^{-3}	6.3×10^{-1}	-
2	2.4×10^{-3}	4.6×10^{-1}	-
3	1.2×10^{-3}	1.2×10^{-1}	-
4	1.3×10^{-2}	1.2×10^{0}	-
5	6.9×10^{-4}	2.2×10^{-1}	-
6	3.9×10^{-3}	6.5×10^{0}	-
7	1.8×10^{-3}	2.8×10^{-1}	-
8	1.2×10^{-3}	2.3×10^{-1}	-
9	5.3×10^{-4}	1.0×10^{-1}	-
10	4.5×10^{-4}	1.4×10^{-1}	-

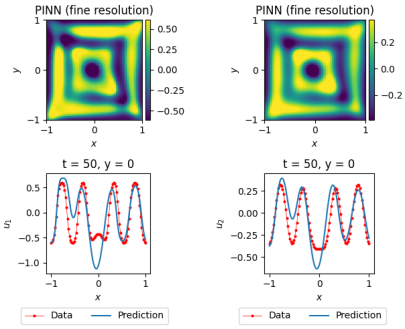


Figure 25: Plots of the diffusion-reaction equation's activator u using PINN trained with finer resolution dataset. *In-dis-test* prediction (left) and *out-dis-test* (right)

on diffusion-reaction increased slightly, however without reaching FINN's accuracy. Identical results were achieved in the Burgers' and diffusion-sorption equations but are omitted due to high conceptual similarity.

D.5. PhyDNet With Original Amount of Parameters

To verify whether our reduction of parameters and the removal of the encoder and decoder layers caused PhyDNet to perform worse, we repeated the experiments using the original PhyDNet architecture as proposed in (Guen & Thome, 2020). However, our results indicate no significant changes in performance, as reported in Table 19, Figure 26, and Figure 27 (bottom). Again, results for the inhibitor u_2 as well as for the Burgers' and diffusion-sorption equations were omitted due to high conceptual similarity.

Table 19: rMSE of PhyDNet using the original network size from ten different training runs for the diffusion-reaction equation.

Run	*Train*	*In-dis-test*	*Out-dis-test*
1	1.2×10^{-3}	1.4×10^{0}	2.2×10^{0}
2	3.9×10^{-4}	5.2×10^{-1}	1.0×10^{0}
3	2.9×10^{-4}	5.4×10^{-1}	1.2×10^{0}
4	6.1×10^{-4}	5.8×10^{-1}	1.1×10^{0}
5	4.6×10^{-4}	5.2×10^{-1}	2.5×10^{0}
6	7.6×10^{-4}	9.8×10^{-1}	3.0×10^{0}
7	4.7×10^{-4}	4.6×10^{-1}	2.6×10^{0}
8	4.7×10^{-4}	4.7×10^{-1}	7.5×10^{-1}
9	3.1×10^{-4}	5.0×10^{-1}	1.2×10^{0}
10	4.7×10^{-4}	4.9×10^{-1}	2.3×10^{0}

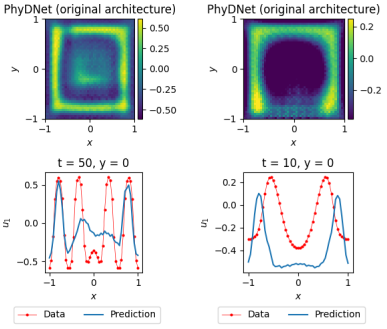


Figure 26: Plots of the diffusion-reaction equation's activator u using PhyDNet with the original network size. *In-dis-test* prediction (left) and *out-dis-test* (right).

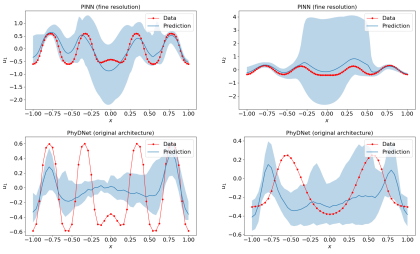


Figure 27: Prediction mean (with 95% confidence interval) of the activator (left) and inhibitor (right) in the diffusion-reaction equation at $t = 50$ compared with the *in-dis-test* data. Top: PINN fine resolution. Bottom: PhyDNet original architecture.

A.4 Publication IV

JAMES | Journal of Advances in Modeling Earth Systems

RESEARCH ARTICLE
10.1029/2023MS004021

Special Collection:
Machine learning application to Earth system modeling

Advancing Parsimonious Deep Learning Weather Prediction Using the HEALPix Mesh

Matthias Karlbauer[1], Nathaniel Cresswell-Clay[2], Dale R. Durran[2,3], Raul A. Moreno[2], Thorsten Kurth[4], Boris Bonev[4], Noah Brenowitz[3], and Martin V. Butz[1]

[1]Department of Computer Science, Neuro-Cognitive Modeling Group, University of Tübingen, Tübingen, Germany, [2]Department of Atmospheric Sciences, University of Washington, Seattle, WA, USA, [3]NVIDIA Corporation, Seattle, WA, USA, [4]NVIDIA Switzerland AG, Zürich, Switzerland

Key Points:

- The model forecasts seven atmospheric variables, an order of magnitude less than that used in state-of-the-art ML weather forecast models
- Forecasts are generated on the HEALPix mesh, facilitating the development of location invariant convolution kernels
- Without converging to climatology, the model produces realistic atmospheric states in 365-day iterative rollouts

Correspondence to:
D. R. Durran,
drdee@uw.edu

Citation:
Karlbauer, M., Cresswell-Clay, N., Durran, D. R., Moreno, R. A., Kurth, T., Bonev, B., et al. (2024). Advancing parsimonious deep learning weather prediction using the HEALPix mesh. *Journal of Advances in Modeling Earth Systems*, *16*, e2023MS004021. https://doi.org/10.1029/2023MS004021

Received 11 SEP 2023
Accepted 25 JUL 2024

Author Contributions:
Conceptualization: Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R. Durran, Martin V. Butz
Data curation: Matthias Karlbauer, Nathaniel Cresswell-Clay, Raul A. Moreno, Thorsten Kurth, Boris Bonev, Noah Brenowitz
Formal analysis: Dale R. Durran
Funding acquisition: Matthias Karlbauer, Dale R. Durran, Martin V. Butz
Investigation: Matthias Karlbauer
Methodology: Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R. Durran
Project administration: Dale R. Durran

Abstract We present a parsimonious deep learning weather prediction model to forecast seven atmospheric variables with 3-hr time resolution for up to 1-year lead times on a 110-km global mesh using the Hierarchical Equal Area isoLatitude Pixelization (HEALPix). In comparison to state-of-the-art (SOTA) machine learning (ML) weather forecast models, such as Pangu-Weather and GraphCast, our DLWP-HPX model uses coarser resolution and far fewer prognostic variables. Yet, at 1-week lead times, its skill is only about 1 day behind both SOTA ML forecast models and the SOTA numerical weather prediction model from the European Center for Medium-Range Weather Forecasts. We report several improvements in model design, including switching from the cubed sphere to the HEALPix mesh, inverting the channel depth of the U-Net, and introducing gated recurrent units (GRU) on each level of the U-Net hierarchy. The consistent east-west orientation of all cells on the HEALPix mesh facilitates the development of location-invariant convolution kernels that successfully propagate weather patterns across the globe without requiring separate kernels for the polar and equatorial faces of the cube sphere. Without any loss of spectral power after the first 2 days, the model can be unrolled autoregressively for hundreds of steps into the future to generate realistic states of the atmosphere that respect seasonal trends, as showcased in 1-year simulations.

Plain Language Summary Weather forecasting traditionally relies on numerical weather prediction models that solve physical equations to simulate the evolution of the atmosphere. Such numerical models are compute intensive, and their performance is increasingly challenged by less compute demanding but still highly sophisticated machine learning (ML) approaches. Yet, a downside for many of these new ML models is they tend to drift away from climatology while producing excessively smoothed fields if they are iteratively stepped forward for several months. Here, a parsimonious machine learning model is developed to forecast just seven atmospheric variables that can be stepped forward to give realistic weather patterns over a full year. Despite using at least a factor of 10 less variables than the 67–227 in the best ML models, our model generates 8-day forecasts with errors that are only a day behind those from state-of-the-art ML forecasts. Our model provides a path toward sub-seasonal and seasonal forecasting that could potentially improve planning for agriculture, water resources, disaster preparedness, and energy production.

1. Introduction

Four years ago, Weyn et al. (2019) posed the question "Can machines learn to predict the weather?" and demonstrated that data driven convolutional neural networks can forecast the evolution of the 500 hPa surface much better than the alternative dynamical model, the barotropic vorticity equation, which was used in the first numerical weather prediction (NWP) model (Charney et al., 1950). An extremely rapid evolution of deep learning weather prediction (DLWP) models followed, culminating in the recent Pangu-Weather (Bi et al., 2023) and GraphCast models (Lam et al., 2022), which outperform the deterministic forecast from the state-of-the-art Integrated Forecast System (IFS) of the European Center for Medium-Range Weather Forecasts (ECMWF).

NWP has continuously improved over the seven decades since the first barotropic model forecast (Benjamin et al., 2019). Current state-of-the-art models typically provide skillful predictions of global weather patterns at effective grid point spacings of roughly 0.1° of latitude (about 10 km) through at least 7 days of forecast lead time (Bauer et al., 2015). The computational effort required to generate such global high-resolution forecasts is enormous and only available at a handful of advanced dedicated centers. Ensemble forecasts, which provide an important way to account for uncertainty by generating a set of equally plausible predictions and extend the limit

AGU ADVANCING EARTH AND SPACE SCIENCES

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Software: Matthias Karlbauer, Nathaniel Cresswell-Clay, Thorsten Kurth
Supervision: Dale R. Durran, Martin V. Butz
Validation: Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R. Durran, Raul A. Moreno
Visualization: Matthias Karlbauer, Nathaniel Cresswell-Clay, Raul A. Moreno
Writing – original draft: Matthias Karlbauer, Dale R. Durran, Martin V. Butz
Writing – review & editing: Matthias Karlbauer, Dale R. Durran, Martin V. Butz

of skillful forecasts beyond that of a single deterministic model run, are also limited by the computational burden of high-resolution NWP to about 50 members (Palmer, 2019).

Global NWP models represent 3D fields as sets of nested spherical shells in which the distance between each shell is the local vertical grid spacing. On every time step, the ECMWF Integrated Forecasting System (IFS), as configured for sub-seasonal forecasting, updates 10 prognostic 3D variables defined at 91 vertical levels. Along with surface pressure, this totals to over 900 spherical shells of data. Here, we use "spherical shell of data" to describe a single variable defined at a single vertical level on a spherical shell covering the globe. The large number of spherical shells of data (combined with the fine horizontal resolution) in NWP models is required to produce acceptably accurate numerical solutions to the equations governing atmospheric motions. The data at each individual point, however, cannot be independently perturbed while maintaining a meteorologically relevant atmospheric state. For example, on horizontal scales larger than about 10 km, the temperatures throughout a vertical column and the heights of constant pressure surfaces must satisfy hydrostatic balance.

The actual number of independent degrees of freedom required to represent the predictable components of the global atmosphere is unknown, but it clearly decreases with increasing forecast lead times (Lorenz, 1969). GraphCast (Lam et al., 2022), for example, has achieved success at lead times as short as 6 hr with 227 spherical shells of data. It can produce forecasts using much less computation time than the ECMWF IFS, but it still requires large computing resources for training: 3 weeks using 32 TPU 4 processors. Pangu-Weather (Bi et al., 2023) cuts the number of spherical shells by almost 2/3 to 69. The spherical Fourier neural operator (SFNO) version of FourCastNet compared with the IFS in Bonev et al. (2023) uses 73 spherical shells of data. Here, we take this reduction much farther, presenting a parsimonious DLWP model that uses just seven spherical shells of data to efficiently provide forecasts approaching the skill of ECMWF. While not as accurate as GraphCast or Pangu-Weather for medium range forecasts with lead times less than 2 weeks, we demonstrate that our model generates far less bias in forecasts of 500-hPa height in 1-year iterative forecasts. In addition, our model is potentially better suited for research applications such as computing the sensitivities of its compact state vector to custom diagnostic functions by backpropagation.

In contrast to many of the recent DLWP architectures, our approach relies on convolutional neural networks (CNN), building on early work by Scher and Messori (2018) and Weyn et al. (2019) and the U-Net configuration in Weyn et al. (2020, 2021). Here, we document substantial improvements over Weyn et al. (2021), obtained by replacing the cubed sphere data representation with the HEALPix mesh, which is widely employed in astronomy (Gorski et al., 2005). In addition, we improve the former model by implementing physically motivated modifications in form of residual connections, recurrent modules, and inverting the channel depth as compared with a standard U-Net.

2. Related Work

Pioneering efforts to create machine learning models to forecast the weather from reanalysis or general circulation model (GCM) output include the dense neural network of Dueben and Bauer (2018) and the CNN models of Scher and Messori (2019) and Weyn et al. (2019), all of which employed latitude longitude (lat-lon) meshes. Weyn et al. (2020) obtained significantly improved forecasts by switching to a cubed sphere mesh with a CNN in the standard U-Net architecture (Ronneberger et al., 2015). Their model was capable of generating realistic weather patterns when stepped forward for a full year (730 12 hr steps). Retaining the cubed sphere, Weyn et al. (2021) produced forecasts out to sub-seasonal time scales using large multi-model ensembles, and Lopez-Gomez et al. (2022) migrated from the U-Net into a U-Net 3+ architecture (Huang et al., 2020)—which adds connections between multiple hierarchical levels in the U-Net—to generate forecasts of extreme surface temperatures.

Returning to the lat-lon mesh, Rasp and Thuerey (2021) demonstrated that a deep Resnet could be pre-trained on GCM data and then fine-tuned by transfer learning on ERA5 data to produce up to 5-day forecasts at coarse 5.65° grid spacing. Building on transformer models from computer vision (Dosovitskiy et al., 2020; Guibas et al., 2021), Pathak et al. (2022) and Kurth et al. (2022) used Fourier neural operators (Li et al., 2020) to develop FourCastNet on a 0.25° lat-lon mesh to generate forecasts approaching the accuracy of ECMWF's IFS. FourCastNet was not, however, capable of stable long-lead-time autoregressive rollouts. This difficulty was overcome by switching from 2D Fourier modes on a lat-lon mesh to spherical harmonic functions Bonev et al. (2023). The resulting SFNO model eliminated much of the vision transformer architecture while improving accuracy and remaining stable for 1-year forecasts.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

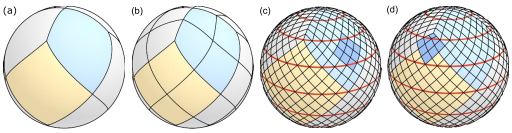


Figure 1. Division of the sphere into 12 faces according to the HEALPix. Four faces to represent either the northern (blue) and southern extratropics, while four more faces arrange around the equator to represent the tropics (yellow). Each face can be subdivided into patches with divisions along the side of each face given by powers of two. The sphere in (a) has a pixel-count of one per face side; we call it hpx1. The sphere in (b) counts two pixels per side (hpx2), whereas the two spheres in (c) and (d) have eight pixels per side, that is, hpx8. Several latitude lines in red emphasize the iso-latitudinal arrangement of the patches. The saturated blue area depicts a 3×3 stencil, as applied by a standard convolution. To apply the 3×3 stencil at the top corner of the equatorial faces, that is, stencil position in (d), we fill in the missing corner patch with the average of the values in the two adjacent patches on the extratropical faces.

Again on a 5.65° lat-lon mesh, Hu et al. (2022) used a shifted window (Swin) transformer (Liu et al., 2021) to produce single forecasts as well as ensembles generated by perturbing the latent state using samples from their learned distribution. Bi et al. (2023) also applied Swin transformers on a lat-lon mesh, but used a fine 0.25° lat-lon grid spacing, 3D transformers, and included latitude and longitude fields as input to train a "3D Earth-specific transformer" at four different forecast lead times of 1, 3, 6, and 24 hr, which are used in combination to span an arbitrary hourly forecast period with minimal model steps. If the ECMWF IFS NWP forecasts are averaged to the coarser 0.25° lat-lon mesh, Pangu-Weather outperforms NWP on several metrics.

In contrast to the preceding approaches, graph neural networks (Battaglia et al., 2018; Gori et al., 2005; Kipf & Welling, 2016; Pfaff et al., 2020; Scarselli et al., 2008) where applied on icosahedral meshes at course resolution by Keisler (2022) and at fine resolution in the GraphCast model (Lam et al., 2022). Similarly to Pangu-Weather, GraphCast appears to outperform the coarsened ECMWF IFS forecast on several metrics.

3. Methods

3.1. Data

3.1.1. Choice of Variables

Beginning with the same six prognostic variables used in Weyn et al. (2021)—geopotential height at 1,000 and 500 hPa (Z_{1000}, Z_{500}), 700–300 hPa thickness ($\tau_{700-300}$) defined as $Z_{300} - Z_{700}$, temperature at 2 m height above ground (T_{2m}), temperature at 850 hPa (T_{850}), and total column water vapor ($TCWV$)—we add Z_{250} based on its importance in the model of Rasp and Thuerey (2021) and to provide an upper tropospheric variable. As in Weyn et al. (2021), three prescribed fields are also provided: topographic height, land-sea mask, and top-of-atmosphere (TOA) incident solar radiation. We do not include prescribed or predicted sea-surface temperature or surface fluxes above the land or ocean. *No* specific information about position on the globe, such as latitude and longitude, is provided. Three-hourly data from the ERA5 reanalysis (Hersbach et al., 2020) provide training data from 1979 to 2012, a validation set from 2013 to 2016, and a test set from 2017 to 2018.

3.1.2. HEALPix Mesh

We discretize all fields using the Hierarchical Equal Area isoLatitude Pixelization (HEALPix) (Gorski et al., 2005). As depicted in Figure 1, a HEALPix mesh is formed by dividing the sphere into twelve equal-area diamond-shaped faces, with four faces lying in the northern and southern hemispheres, and four in the tropics. According to Gorski et al. (2005), the HEALPix mesh has three important properties. *(a) Hierarchical structure of the database*: Each of the 12 base faces can be progressively subdivided into smaller patches. *(b) Equal areas for the discrete elements of the partition*: All patches are the same size. *(c) Isolatitude distribution for the discrete area elements on the sphere*: The patches line up with lines of latitude, facilitating the computation of zonal averages and one-dimensional zonal spectra. Importantly, this last property makes the HEALPix mesh an "east is to the right" grid, which facilitates the training of a single set of position invariant convolutional kernels to capture the motion of typical weather disturbances, as discussed in Section 4.1.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

The HEALPix can be considered a graph and does not allow a seamless application of convolution operations. Thus, Perraudin et al. (2019) explicitly define a graph from the HEALPix—by connecting adjacent neighbors with weighted edges—and perform a graph convolution to classify weak lensing maps from cosmology. In a different approach, Krachmalnicoff and Tomasi (2019) classify digits and determine cosmic parameters from simulated cosmic microwave background maps. They apply 1D convolutions to the flattened HEALPix data with a kernel size k and stride s both equal to 9, appending a zero to those cases where only seven instead of eight neighbors are defined (top corner of the tropical faces). In contrast, we treat the 12 faces as distinct images and pad their boundaries using data from neighboring faces to allow the computation of 2D convolutions and averaging operators directly, as detailed in Appendix A. To accelerate the padding operation, we have implemented a custom CUDA kernel, which is available in our repository.

The grid spacing, or shortest inter-node spacing, on the HEALPix mesh is the diagonal distance between a pair of nodes on adjacent latitude lines. Denoting a HEALPix mesh with n divisions along one side of the original 12 faces as HPXn. The grid spacing is approximately 220 km (≈2°) for HPX32 and 110 km (≈1°) for HPX64.

3.2. Machine Learning Architecture

Relating to Tobler's first law of geography: "All things are related, but nearby things are more related than distant things." (Tobler, 1970), we mostly retain the comparably simple U-Net structure from Weyn et al. (2020). U-Nets (Ronneberger et al., 2015) are hierarchically structured feed-forward convolutional neural networks that were originally proposed for segmenting biomedical images. The U-Net structure proposed here introduces several physically motivated advancements to the vanilla U-Net used by Weyn et al. (2021) for sub-seasonal forecasting. Our final model configuration is visualized as a sequence of operations on layers or a block of layer operations in Figure 2. The latter case is indicated by CNB or GRU, which refer to `ConvNeXt`- and `GRU`-blocks (cf., Sections 3.2.1 and 3.2.3 for explanations), respectively. Details of the `ConvNeXt`-block structure are also visualized. GRU-blocks augment the respective layer with a recurrent processing mechanism (cf., Section 3.2.3). Table 1 specifies the respective parameter settings. Color codes for the operations in Table 1 approximate those used in the model schematic in Figure 2. For example, the operations in red are 3×3 convolutions followed by GELU activation functions. Residual connections are only reported in Table 1 if they contribute to the parameter count when implementing a 1×1 convolution to adjust the channel depth. In the following, we describe the incremental advancements that we add to our model.

3.2.1. Residual Prediction

We switch to a residual prediction approach both for the full predictive step and within each ConvNeXt block. The ConvNeXt block (Liu et al., 2022) is designed to minimize compute, while maintaining performance. It introduces an inverted channel-bottleneck where the kernel size is reduced to $k = 1$. This saves parameters and compute, because channel depth is only processed with a 1×1 spatial filter. As shown in Figure 2, though, we modify the original ConvNeXt block from Liu et al. (2022) by implementing a kernel size of $k = 3$ and employing a two-stage convolution as done in Weyn et al. (2021).

Predicting residuals, that is, changes over a time step, instead of full values, is similar to the discretization of time derivatives when solving partial or ordinary differential equations, and has been used successfully in previous DLWP models (Hu et al., 2022; Keisler, 2022; Lam et al., 2022; Pathak et al., 2022).

3.2.2. Inverting the Ordering of Channel Depth

The standard U-Net for semantic segmentation (Ronneberger et al., 2015) and its successors (Huang et al., 2020; Zhou et al., 2018) employ relatively few channels on the highest level and successively double the channel depth, while halving the spatial resolution in each deeper layer. This ordering is useful in image segmentation tasks, where deeper channels are required to create increasingly abstract filters to identify semantic features and express complex objects. In weather prediction, however, we find it is better to devote more capacity to the layers in the first level, where a wide variety of fine grained weather phenomena must be captured. Deeper layers at coarser resolution, on the other hand, need only encode larger scale atmospheric motions, which can be adequately represented with comparably fewer channels.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

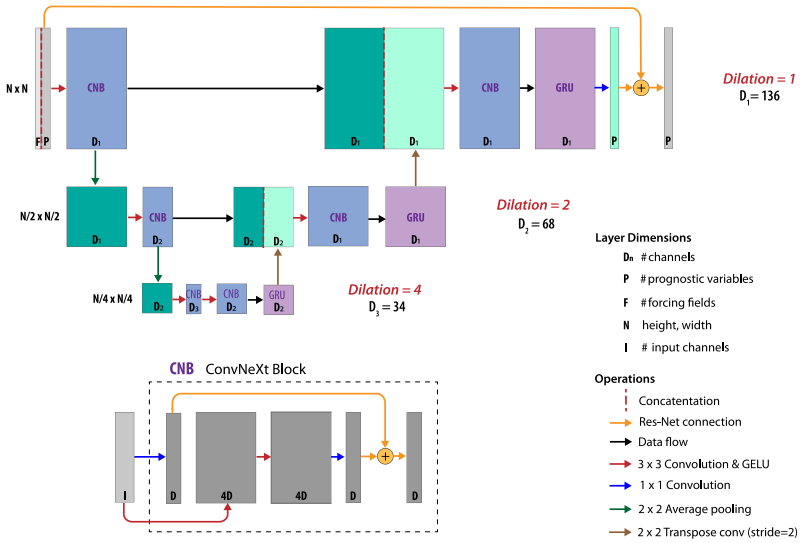


Figure 2. Schematic representation of our DLWP-HPX architecture as a sequence of operations on layers (see legend). Individual layers are labeled by their channel depth, with $D_1 = 136$, $D_2 = 68$, and $D_3 = 34$ being associated with the first convolutions in each of the three U-Net levels. Each ConvNeXt block (blue) is replaced by the layers and operations shown in the inset labeled CNB, with generic depths D and I determined by the channel depth of the input and the labeled value of D_n. The purple blocks labeled GRU denote convolutional Gated Recurrent Unit layers, which are augmented with 1×1 spatial convolutions. Other layers evaluated by the encoder are shown as dark green, while those evaluated by the decoder are shown as light green.

Thus, we invert the channel order, employing 136, 68, and 34 channels in each convolution on the first, second, and third layer, respectively (cf., Figure 2). While this modification improves the model performance significantly, it also increases the computational burden, since more computations and data processing are required to evaluate the additional convolutions at fine spatial resolution. Tests which preserved the total number of trainable parameters, but completely eliminated the deeper layers in the U-Net gave worse results, demonstrating that the longer-range connections and richer latent space structures enabled by the full U-Net architecture remain important.

3.2.3. Recurrent Modules

The vanilla U-Net is a feed-forward network, which treats successive inputs independently even if the data represents a continuous sequence over time. Feed-forward networks do not have any memory capacity. They do not maintain an internal state between time steps. To exploit information from previous latent states, we include a gated recurrent unit (GRU) (Cho et al., 2014) at the end of each decoder block, implemented as a convolutional GRU (Ballas et al., 2015) with 1×1 spatial convolutions. GRUs use a hidden latent state that accumulates information over time to influence the current forecast step. We chose GRUs over LSTMs (Hochreiter & Schmidhuber, 1997) since we re-initialize the recurrent data over each 24-hr cycle, and therefore do not require forget-gates (as confirmed experimentally, not shown).

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Table 1
CNN Architecture as a Sequence of Operations on Layers; c_{in}, k, s, and d Denote the Number of Input Channels, Kernel Size, Stride, and Dilation

Layer	c_{in}	k	s	d	Receptive field	Output shape	Parameter count		
							Weights	Biases	Total
ConvNeXt									
Conv2d	18	1	1	1	1 × 1	(12, 64, 64, 136)	2,448	136	2,584
Conv2d	18	3	1	1	3 × 3	(12, 64, 64, 544)	88,128	544	88,672
Conv2d	544	3	1	1	5 × 5	(12, 64, 64, 544)	2,663,424	544	2,663,968
Conv2d (1)	544	1	1	1	5 × 5	(12, 64, 64, 136)	73,984	136	74,120
AvgPool2d	136	2	2	—	6 × 6	(12, 32, 32, 136)	0	0	0
ConvNeXt									
Conv2d	136	1	1	1	6 × 6	(12, 32, 32, 68)	9,248	68	9,316
Conv2d	136	3	1	2	14 × 14	(12, 32, 32, 272)	332,928	272	333,200
Conv2d	272	3	1	2	22 × 22	(12, 32, 32, 272)	665,856	272	666,128
Conv2d (2)	272	1	1	1	22 × 22	(12, 32, 32, 68)	18,496	68	18,564
AvgPool2d	68	2	2	—	24 × 24	(12, 16, 16, 68)	0	0	0
ConvNeXt									
Conv2d	68	1	1	1	24 × 24	(12, 16, 16, 34)	2,312	34	2,346
Conv2d	68	3	1	4	56 × 56	(12, 16, 16, 136)	83,232	136	83,368
Conv2d	136	3	1	4	88 × 88	(12, 16, 16, 136)	166,464	136	166,600
Conv2d	136	1	1	1	88 × 88	(12, 16, 16, 34)	4,624	34	4,658
ConvNeXt									
Conv2d	34	1	1	1	88 × 88	(12, 16, 16, 68)	2,312	68	2,380
Conv2d	34	3	1	4	120 × 120	(12, 16, 16, 136)	41,616	136	41,752
Conv2d	136	3	1	4	152 × 152	(12, 16, 16, 136)	166,464	136	166,600
Conv2d	136	1	1	1	152 × 152	(12, 16, 16, 68)	9,248	68	9,316
GRU									
Conv2d	136	1	1	1	152 × 152	(12, 16, 16, 136)	18,496	136	18,632
Conv2d	136	1	1	1	152 × 152	(12, 16, 16, 68)	9,248	68	9 316
ConvTrans2d	68	2	2	1	154 × 154	(12, 32, 32, 68)	18,496	68	18,476
Concat (2)	—	—	—	—	—	(12, 32, 32, 136)	0	0	0
ConvNeXt									
Conv2d	136	3	1	2	154 × 154	(12, 32, 32, 272)	332,928	272	333,200
Conv2d	272	3	1	2	162 × 162	(12, 32, 32, 272)	665,856	272	666,128
Conv2d	272	1	1	1	170 × 170	(12, 32, 32, 136)	36,992	136	37,128
GRU									
Conv2d	272	1	1	1	170 × 170	(12, 32, 32, 272)	73,984	272	74,256
Conv2d	272	1	1	1	170 × 170	(12, 32, 32, 136)	36,992	136	37,128
ConvTrans2d	136	2	2	1	171 × 171	(12, 64, 64, 136)	73,984	136	74,120
Concat (1)	—	—	—	—	—	(12, 64, 64, 272)	0	0	0
ConvNeXt									
Conv2d	272	1	1	1	171 × 171	(12, 64, 64, 136)	36,992	136	37,128
Conv2d	272	3	1	1	173 × 173	(12, 64, 64, 544)	1,331,712	544	1,332,256

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Table 1
Continued

Layer	c_{in}	k	s	d	Receptive field	Output shape	Parameter count		
							Weights	Biases	Total
Conv2d	544	3	1	1	175 × 175	(12, 64, 64, 544)	2,663,424	544	2,663,968
Conv2d	544	1	1	1	175 × 175	(12, 64, 64, 136)	73,984	136	74,120
GRU									
Conv2d	272	1	1	1	175 × 175	(12, 64, 64, 272)	73,984	272	74,256
Conv2d	272	1	1	1	175 × 175	(12, 64, 64, 136)	36,992	136	37,128
Conv2d	136	1	1	1	175 × 175	(12, 64, 64, 14)	1,904	14	1,918
							9,816,752	6,066	9,822,818

Note. Output shape is face × height × width × channels. Dashed line separates model's encoder (above) and decoder (below). "Concat" implements skip connections by appending the state in parenthesis, numbered earlier, to the output of the previous layer. The result from the orange 1 × 1 convolution at the beginning of most ConvNeXt blocks is added to the corresponding output channel to form a residual connection.

3.2.4. Miscellaneous Modifications

Several other components of the original Weyn et al. (2021) model were modified based on recent results from deep learning research: The capped leaky ReLU was replaced by GELU activation functions with a cap at 10 (Hendrycks & Gimpel, 2016). The GELU has a smooth derivative at zero, which facilitates the optimization of deep learning models. Upsampling was changed from nearest-neighbor sampling (knn-sampling with $k = 1$) to a transposed convolution. Finally, the pairs of two successive convolutions were replaced at each encoder and decoder level in the U-Net by a modified ConvNeXt block (Liu et al., 2022), as visualized in Figure 2.

3.2.5. Time Stepping Scheme

Similarly to Weyn et al. (2021), we apply a two-in-two-out mapping with a temporal resolution twice as fine as the actual time step. For example, two atmospheric states 3 hr apart (each consisting of seven prognostic, along with three prescribed fields) are concatenated and input to the model, which generates a new pair of states, each characterizing the atmosphere 6 hr later in time. This strategy is observed to stabilize and accelerate the training, since the model receives additional information about the atmosphere's rate of change and only has to be called half as often.

The frequency spectrum of atmospheric kinetic energy has a strong peak at 24 hr because many circulations are modulated by solar heating. We therefore evaluate the training loss function as the mean squared error over a 24-hr period. Tests in which the MSE was evaluated over multi-day periods tended to result in a model that gradually approached climatology over many recursive steps.

Training our model only over one daily cycle does mean that the recurrent states of the GRUs are not optimized for long rollouts. To prevent the explosion of recurrent states when generating long multi-day forecasts, we re-initialize the recurrent states every 24 hr as illustrated in Figure 3 for a 12-hr time step with 6 hr resolution. For training or for the first step in a long forecast rollout, the model predicts $\left[\hat{s}_{(t+6)}, \hat{s}_{(t+12)}\right]$ from initial data $[s_{(t-6)}, s_{(t)}]$, and then in the subsequent step uses $\left[\hat{s}_{(t+6)}, \hat{s}_{(t+12)}\right]$ to predict $\left[\hat{s}_{(t+18)}, \hat{s}_{(t+24)}\right]$. But before this, the hidden states for the GRUs are initialized in a preliminary step by calling the model once with the state pair $[s_{(t-18)}, s_{(t-12)}]$ and a hidden state h_0 initialized with zeros. The resulting forecast for $\left[\hat{s}_{(t-6)}, \hat{s}_{(t)}\right]$ is discarded, but the hidden state h_1 is supplied to the GRU and paired with the actual initial data $[s_{(t-6)}, s_{(t)}]$ for the first step of the model. As shown by the bottom row in Figure 3, in a forecast rollout, the next day's prediction begins by re-initializing the GRU starting with forecast values from one time step earlier and h_0 set to zero to obtain h_1. Note that since the GRU is re-initialized every day, there would be five model steps per day when using a 6 hr time step (with 3 hr data resolution).

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

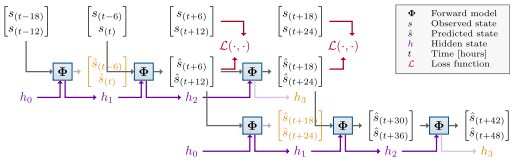


Figure 3. Two time-level input-output scheme with GRU for training and inference assuming 6 hr time resolution. The output from the preliminary initialization step (in orange) is discarded, but the hidden state h_1 is generated and used in the first model step. The hidden state h_3 (in orange) at the end of the 24 hr forecast is discarded as the GRU will be re-initialized for the next recursive inference step (lowest row). For training (top right), the loss function is computed from the four forecast times spanning a 24 hr period at 6 hr resolution, as indicated in red.

3.2.6. Training

Our best performing DLWP-HPX model, described above, has 9.8 M parameters that are trained for 300 epochs (equivalent to 931,199 update steps) over 8 days on four NVIDIA A100 GPUs with 80 GB VRAM each. A batch size of eight per GPU is chosen, effectively resulting in an overall batch size of 32. We combine the Adam optimizer (Kingma & Ba, 2014) with a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016), setting the initial learning rate to 2×10^{-4} and gradually refining it to zero. To stabilize the training, we clip the gradients to the current learning rate, which we observe to be particularly beneficial for large recurrent models.

3.3. The Receptive Field

Several leading DLWP models (Bi et al., 2023; Chen et al., 2023; Hu et al., 2022; Pathak et al., 2022) are based on Vision Transformers (ViTs) (Dosovitskiy et al., 2020), which were originally developed to account for non-local relationships in images; effectively working on patch embeddings. ViTs are successors of Transformers (Vaswani et al., 2017), which were introduced to efficiently accommodate very non-local relationships in natural language processing (NLP), where no fixed upper bound exists on the distance between words that may interact to change the meaning of a sentence. In contrast to ViTs, we use a U-Net to emphasize local atmospheric interactions, nevertheless each step of our model samples from a very large receptive field. (The "receptive field" is the set of grid cells the model accesses when generating output for a specific target pixel.)

There is a strong physical constraint on the locality of atmospheric interactions, which is that *no atmospheric disturbances travel faster than the speed of sound*, roughly 300 m/s. Sound waves are not meteorologically significant, and are not represented in the data used to train ML weather models. A better measure of the speed of the fastest moving signals of meteorological importance is the transport by the strongest jet-stream winds, which could transport a passive tracer at roughly 100 m/s, or about 4,300 km in 12 hr.

The pair of 2×2 average poolings and the dilations in the second and third levels of our U-Net architecture (Figure 2) substantially widen the receptive field that potentially influences the solution at a given point after each forward step of our model. Neglecting influences from special points at the corners of the 12 basic HEALPix faces, the receptive field at each stage of the neural network is listed in Table 1 and grows to a 175×175 patch of cells after the last 3×3 convolution in the decoder.

The diagonal distance between adjacent points on our 3×3 stencil (dark blue patch in Figure 1) on a HPX64 mesh is approximately 110 km. Thus, the receptive field for one step of our full HPX64 model is a patch exceeding 18,900 km on each side, which is large enough to include all points influenced by sound wave propagation over a 12 hr time step, and far more than would be required to contain the fastest moving meteorologically significant signals present in the ERA5 training data. In particular, at every step, our HPX64 forecast at a given point is influenced by a set of surrounding points containing roughly 70% of all the cells covering the globe.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Table 2
Number of Trainable Parameters in Millions, Number of Spherical Shells of Prognostic Variables, Horizontal Resolution in Degrees Latitude, and Temporal Resolution (Δ_t) of the Models Compared in Figure 4

Model	Parameters	Spherical shells	Resolution	Δ_t
Weyn 2021	2.7 M	6	1.4°	6 hr
Our HPX64	9.8 M	7	1°	3 hr
ECMWF	—	900+	0.15°	0.2 hr
GraphCast	21 M	227	0.25°	6 hr

4. Results

In the following, we first evaluate key variables in our model over a 14-day forecast lead time, which is slightly longer than the period over which knowledge of the initial atmospheric conditions gives these single deterministic forecasts some predictive skill. We compare our best model with the ECMWF S2S forecasts and with our previous Weyn et al. (2021) results. We then document the successive improvements that our changes in model architecture have on the RMSE and ACC scores for Z_{500}. Next, we examine the ability of the model to distinguish between the amplitudes of the daily T_{2m} ranges in tropical forests, in deserts, and over the ocean. Finally, we examine the behavior of the simulations over sub-seasonal (8-week) and 1-year free running rollouts.

4.1. Quantitative Performance Through 14-Day Forecast Lead Time

To compare our model with the results from Weyn et al. (2021) and to state-of-the-art NWP from ECWMF, we compute both root mean squared error (RMSE) between observations and model predictions and anomaly correlation coefficient (ACC) scores with respect to the ERA5 climatology. Both metrics are compared on a $1° \times 1°$ lat-lon mesh and weighted by latitude, requiring us to project our DLWP-HPX and Weyn et al. (2021) forecasts from the HEALPix and cubed sphere meshes onto the lat-lon grid. Because our ultimate focus is on sub-seasonal and seasonal forecasting, we compare against ECMWF's integrated forecasting system for sub-seasonal forecasts (IFS S2S), which were initialized bi-weekly on Mondays and Thursdays and stepped forward at about 16 km effective resolution for the first 15 days (then doubling to 32 km) (ECMWF, 2024a). For comparison with Weyn et al. (2021), our test set focuses on the years 2017 and 2018. In this and all the following cases, except a few simulations in our ablation study, computations are performed at HPX64 and 3 hr resolution (corresponding to 6 hr time steps).

To further compare our model with a state-of-the-art DLWP model, we include Z_{500} scores for GraphCast, retrieved from the interactive WeatherBench2 (Rasp et al., 2023) homepage (Google, 2024b). In contrast to the others, GraphCast scores are computed on its native $0.25° \times 0.25°$ grid and for 2018 only, since the model was trained on data including 2017. Key parameter attributes of the model from Weyn et al. (2021), IFS S2S, GraphCast, and our HPX64 model are listed in Table 2.

The GraphCast-WeatherBench2-RMSE scores at T_{850}, and particularly at T_{2m}, are difficult to compare with those from our model at early forecast lead times because differences in resolution and grid structure influence the representation of the topography and coastlines. Therefore we only plot GraphCast scores at Z_{500}. As previously documented, the RMSE and ACC of GraphCast temperature forecasts at $0.25° \times 0.25°$ resolution, are somewhat better than those from the IFS (Lam et al., 2022).

As shown in Figure 4, the RMSE scores for Z_{500}, 24-hr-averaged T_{2m} (because instantaneous T_{2m} fields are not archived from the ECMWF S2S forecasts (ECMWF, 2024b)), and T_{850} all improve substantially compared to Weyn et al. (2021). Moreover, despite the small number of prognostic variables and coarse spatial resolution of our model, the RMSEs for Z_{500} only lag the scores for ECMWF S2S and GraphCast by about 1 day at 1-week lead time. The HPX64 RMSE for T_{850} shows a similar lag in skill compared to the IFS. As expected theoretically, the RMSE scores for all models appear to be asymptotically approaching $\sqrt{2}$ times climatology beyond 2 weeks when the skill of a single deterministic forecast drops toward zero. We present the comparison of 24-hr-averaged T_{2m} between our model and IFS S2S for completeness, but it should be interpreted with caution. The re-gridding of both the IFS S2S and the HEALPix data to the $1° \times 1°$ lat-lon analysis grid introduces errors in the representation of coastlines and topography that significantly influence the surface temperature field. As a consequence, the RMSE values shown in Figure 4b are not representative of those in each model's native representation of the T_{2m} field.

One additional issue that arises when plotting initial RMSE (and to a lesser extent ACC) for the ECMWF IFS S2S model is that, unlike our DLWP-HPX model, the IFS forecasts are not initialized with the ERA5 data. Thus, at very short forecast lead times, differences between the IFS initialization and the ERA5 data introduce apparent errors in the IFS forecast that are not representative of its actual performance. Lam et al. (2022) accounted for this in their comparison between the IFS and GraphCast, but it requires considerable extra computation. We are not

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

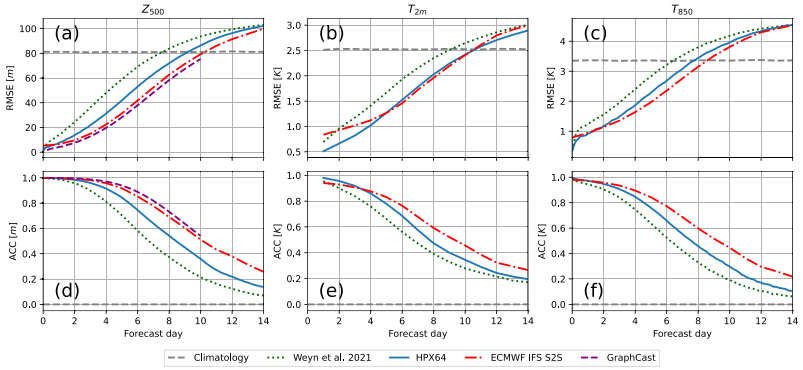


Figure 4. Comparison of the performance of the DLWP-HPX, Weyn et al. (2021), ECMWF IFS S2S, and GraphCast models. GraphCast is averaged over 104 forecasts for 2018, while other forecasts are averaged over 204 forecasts from 2017 through 2018. RMSE for (a) Z_{500}, (b) T_{2m}, and (c) T_{850}; climatology is indicated by the gray dashed line. ACC for or (d) Z_{500}, (e) T_{2m}, and (f) T_{850}.

claiming to outperform the IFS, so we simply suggest using caution when comparing errors between our models and the IFS at lead times less than 2 days.

ACC scores for Z_{500}, T_{2m}, and T_{850} are also shown in Figures 4d–4f. As with RMSE, there is substantial improvement relative to both the previous model from Weyn et al. (2021) and the IFS S2S. In meteorological contexts, an ACC score of 0.6 is typically considered the lower limit of practical skill. The scores from our HEALPix model cross this threshold at about 7.5 days for Z_{500} and 6.5 days for T_{850}, both of which are about 1.5 days sooner than the respective results for the IFS S2S and for the GraphCast Z_{500} forecast. Numerical comparisons of the model RMSE and ACC scores averaged over the same 208 forecasts used to plot Figure 4 are given for 3- and 5-day lead times in Table 3.

The relative importance of the various improvements in model architecture between Weyn et al. (2021) and our best DLWP-HPX model is illustrated for the Z_{500} field in Figure 5. The total number of trainable parameters is held constant at roughly 2.7×10^6 over the first five sets of changes. The RMSE rises to 50 m around 4.2 days in Weyn et al. (2021) (dark green dotted curve); replacing the 64×64 cubed sphere by a HPX32 grid (aqua curve) delays the error growth by about 0.5 days despite the associated 50% reduction in total grid points. There is also a similar substantial improvement in the ACC. Continuing with the HPX32 mesh, we replace the capped ReLU by a capped GELU activation function, replace knn-interpolation by strided transposed convolution, and introduce

Table 3
Root Mean Squared Errors (RMSE) and Anomaly Correlation Coefficient (ACC) Scores for Weyn et al. (2021) (W21), Our HPX64, and ECMWF's IFS Models, Evaluated on Geopotential at 500 hPa (Z_{500}), Temperature 2 m Above Ground (T_{2m}), and Temperature at 850 hPa (T_{850}) on Lead Times of 3 and 5 Days

		Z_{500}			T_{2m}			T_{850}		
Lead time		W21	HPX64	IFS	W21	HPX64	IFS	W21	HPX64	IFS
RMSE	3 days	36.26	21.88	14.91	1.17	0.82	1.02	1.95	1.49	1.35
	5 days	59.01	41.91	31.30	1.67	1.27	1.27	2.83	2.28	1.96
ACC	3 days	0.90	0.96	0.98	0.84	0.92	0.91	0.84	0.91	0.94
	5 days	0.70	0.84	0.92	0.66	0.78	0.83	0.64	0.76	0.84

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

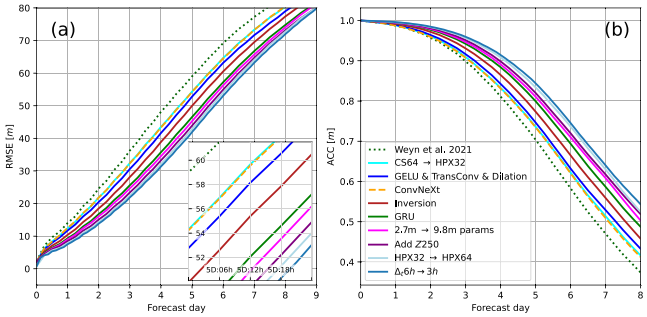


Figure 5. Impact of successive model improvements on the accuracy of Z_{500} RMSE. Each successive change builds on top of the previous architecture, adding the modification indicated in the legend: (a) RMSE, (b) ACC. Inset in (a) provides a magnified view of the error growth between 5 and 6 forecast days.

dilated convolutions in the 2 lower levels of the U-Net (as detailed in Figure 2); this yields the modest but distinct improvements shown by the dark-blue curves.

Next, we replace the pairs of convolutions in each level of the encoder and decoder by a ConvNeXt block with kernel size $k = 3$ (dashed tan curve). This actually produces a slight degradation in performance, but in other configurations closer to our final model, the ConvNeXt block does improve the performance, and importantly, it also reduces the memory footprint by about 25% at a constant parameter count. A further significant improvement is obtained by inverting the standard U-Net progression in channel depth to have the most channels at the highest spatial resolution and the fewest at the lowest resolution (dark red curve). The final significant improvement in the 2.7-million parameter model is obtained by adding recurrence in the form of GRU cells in the decoder (green curve).

After adding the GRU cells, the rise of the RMSE to 50 m is delayed to about 5.3 days and the drop of the ACC below 0.6 to roughly 6.8 days. The next series of changes produces successive small improvements that push these values out to about 5.7 days for RMSE and 7.4 days for ACC. These improvements, as sequentially plotted in Figure 5, are: increasing the number of trainable parameters to 9.8×10^6, adding the Z_{250} field, increasing the horizontal resolution to HPX64 (which is more important for ACC than RMSE particularly on T_{2m}), and decreasing the time resolution to 3 hr. Benefits from the use of 3-hr time resolution were only obtained if the model was configured with the GRUs.

The single most effective modification in the preceding set of successive improvements is the migration from the cubed sphere to the HEALPix mesh, even though the 64×64 cubed sphere has twice the total number of grid-points as the HPX32 mesh. A likely explanation for the superiority of the HEALPix mesh is not simply that it is a more uniform covering of the globe than that provided by the cubed sphere, but that it allows us to train a single set of location-invariant kernels for use over the entire globe. Note that east and west have the same orientation in every HEALPix cell; we refer to this property as "east to the right." In particular, the center and the east and west corners of each HEALPix cell are all at the same latitude. (A similar relationship holds in the north-south direction for meridians passing through those cells lying equatorward of the maximum north-south extent of the four equatorial faces in Figure 1a.) Thus, on the HEALPix mesh, eastward motion at all points and at all latitudes would be in the same direction across the diamond-shaped 3×3 stencil in Figure 1c. In contrast, at any point on either of the polar faces on the cubed sphere, east could map to any of four directions along the axes of the 3×3 convolutional stencil, depending on its longitude, as visualized in Appendix A.

Since most large-scale weather systems move in a generally eastward direction in mid and high latitudes, we believe the "east-to-the-right" property allows a fixed number of kernel elements to more efficiently produce the

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

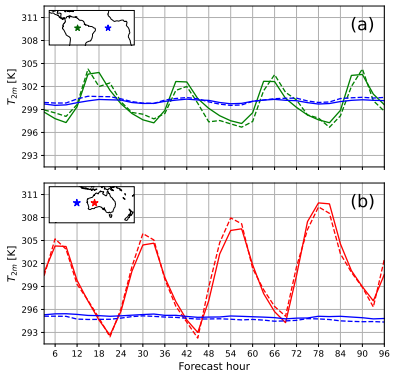


Figure 6. HPX64 simulation of the diurnal cycle of T_{2m} (solid curves) at the four locations shown in the insets starting from 00 UTC on 12 March 2018. ERA5 values for the same $1° \times 1°$ lat-lon cell are shown as dashed lines. Values are plotted every 3 hr.

required set of flow evolutions in the latent layers. This is because we can train one set of kernels for use everywhere on the HEALPix mesh instead of training separate sets of kernels for the equatorial and for the polar faces on the cubed sphere (Weyn et al., 2021). A HEALPix model with the same total number of trainable parameters as the cubed sphere model can, therefore, employ twice as many trainable elements within each kernel.

4.2. Eliminating the Need for Boundary-Layer Parameterizations

Accurate forecasts of surface temperatures in NWP models rely on the empirical parameterization of multi-scale processes near the Earth's surface in the atmospheric boundary layer (ABL). The bottom of the ABL includes the roughness layer (2–5 times the height of roughness elements such as vegetation), and the surface layer (often 10–100 m deep), where shear-driven turbulence dominates generation by convection. The depth of the full ABL, where larger-scale eddies and circulations communicate the processes in the surface layer to the free atmosphere, can vary from $O(100)$ m in calm stable nighttime conditions to several kilometers during the day over deserts.

No effort is made to explicitly account for ABL processes in our model; the T_{2m} field is treated the same as the other six prognostic fields. The same CNN kernels are employed everywhere over the globe on the HEALPix mesh; the only data that might distinguish one location from another are the land-sea mask, the terrain elevation, and the TOA solar forcing; neither longitude nor latitude are provided. Yet our model does a good job of capturing the diurnal cycle in multi-day forecasts over very different surfaces. Figure 6 shows the diurnal cycle in T_{2m} at locations over the Amazon forest, the Australian desert, and two adjacent oceans over a 4-day simulation starting at 00 UTC on 12 March 2018.

Compared to over land, the diurnal T_{2m} variations are modest over the oceans, and they are well captured by our model. The land-sea mask is undoubtedly important in distinguishing the ocean locations from those over land. More interestingly, the model does an excellent job of capturing the large diurnal temperature range over the Australian desert, while correctly generating a much lower amplitude signal over the Amazon. The prognostic field that has most likely facilitated this distinction is *TCWV*, which is significantly higher over the Amazon than over the Australian desert. The model also captures the 4-day trend for increasing temperatures over Australia, which is linked to the evolution of larger-scale weather systems. Overall, the ability of the model to capture the diurnal T_{2m} cycle with just seven prognostic fields, without any special treatment of the ABL, and without geo-specific inputs such as latitude and longitude is suggestive of the power and potential of DLWP-HPX.

4.3. Iterative Rollouts Over Subseasonal to Annual Time Scales

There are three time scales of primary interest for global atmospheric simulations: medium-range weather forecasting for lead times of up to 2 weeks, sub-seasonal and seasonal forecasts for lead times up to 6–9 months, and climate simulations over periods of tens to hundreds of years. Our focus is on the sub-seasonal to seasonal time scale; therefore, in this section we examine the model's performance in iterative rollouts over periods up to 1 year.

To investigate the stability and drift in model simulations over a full annual cycle, we initialize it using ERA5 data for 00 UTC on 1 June 2017 (together with the 21 UTC fields on 31 May). Using 6-hr time steps (with 3-hr time resolution), we perform 1,460 iterations to generate a 365-day simulation. The 3-day running mean of Z_{500}, averaged around each latitude, is plotted as a function of latitude and time in Figure 7, along with the corresponding averages from the ERA5 data. Despite being trained to minimize RMSE over a single day and not enforcing any physical constraints, the DLWP-HPX simulation responds to the TOA solar forcing to generate the annual cycle reasonably well.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

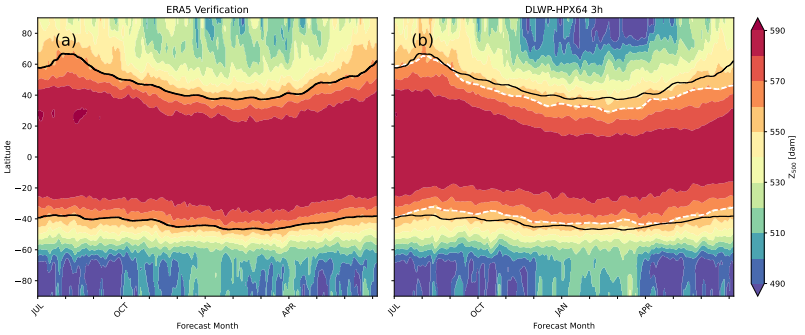


Figure 7. Zonally averaged 3-day mean of Z_{500} plotted as a function of time and latitude for 1 year beginning on 1 July 2017 for: (a) the ERA5 reanalysis, and (b) a recursive 1-year rollout of the DLWP-HPX model. Also shown are 15-day averaged values of the 5,600 m contour of Z_{500} for the ERA5 data (black lines) the DLWP-HPX simulation (white dashed lines).

One region where the errors are significant is the arctic. About 5 months into the simulation, the simulated heights in the arctic region drop as much as 60 m below those in the reanalysis during the boreal winter. In contrast, at 5–8-month lead times, the heights in the antarctic region increase to approximately correct values in the austral summer. The asymmetry between the response in arctic and antarctic flips if the 1-year rollout begins 6 months later. When the simulation is initialized on 2 January 2018, the heights in the arctic during boreal winter are approximately correct, while those in the antarctic are too cold (Figure 8d).

There is also a long-term drift toward lower heights in the subtropics and mid-latitudes, creating a roughly 30 m loss in Z_{500} by the end of the 1-year forecast. The 30-m loss amounts to 0.5% of the full Z_{500} value and to 8.7% of the Z_{500} standard deviation (computed from the reanalysis data of the forecasted period). Climate models are

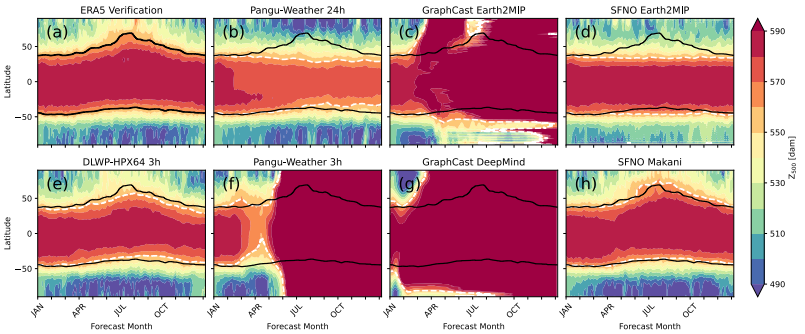


Figure 8. Zonally averaged 3-day mean of Z_{500} plotted as a function of time and latitude: (a) for ERA5 reanalysis, (b)–(h) for recursive 1-year simulations for each model as identified in the titles, initialized on 2 January 2018. Also shown are 15-day averaged values of the 5,600 m contour of Z_{500} for the ERA5 data (black lines) each model simulation (white dashed lines).

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

tuned to avoid long-term drift in the predicted fields, but operational NWP models are not so tuned. For example, significant model biases that grow over a time scale of several weeks are removed to create sub-seasonal ECMWF IFS S2S forecasts (Vitart, 2004; Weigel et al., 2008). To facilitate comparison of model drift with the ERA5 reanalysis, the pair of black lines in both panels show the 15-day mean of the zonally averaged 560-dam Z_{500} contours in the northern and southern hemisphere. The white lines in Figure 7b show the corresponding 560-dam Z_{500} contours for the DLWP-HPX simulation. The drift toward lower heights starts to become evident after 2 months in the northern hemisphere and continues to grow slowly for the remainder of the year. Differences show up earlier in the southern hemisphere, but the average drift is smaller and even disappears at a few times later in the year. As will be discussed in a forthcoming paper, both the errors near the poles and the drift in the tropics in Z_{500} can be corrected by incorporating SST forecasts from a coupled atmosphere-ocean model.

The performance of three additional state-of-the-art DLWP models is compared with our model using this same metric in Figure 8, which shows the evolution of zonally averaged Z_{500} heights over a 1-year rollout beginning 2 January 2018. This year is part of the test set for all of the models: our DLWP-HPX, Pangu-Weather, GraphCast, and FourCastNetv2 based on spherical Fourier neural operators (SFNO) (Bonev et al., 2023). Details about the code used to generate these rollouts can be found in the Data Availability Statement (i.e., Appendix A2 in Appendix A).

The Pangu-Weather model does not include solar forcing, and therefore, it does not follow the annual cycle. When stepped forward with a 24-hr time step (Figure 8b), significant drift is apparent after about 1.5 months, which grows through the year without pushing the simulation into grossly unrealistic states. Based on the discussion of Extended Data, Figure 7a in Bi et al. (2023), one would not expect good performance from Pangu-Weather if rolled out with a 3-hr time step, and indeed the 3-hr rollout starts to produce significant errors after 1.5 months and generates completely unrealistic results after about 5 months (Figure 8f). We nevertheless, show its performance to contrast it with our 3-hr-time-resolution rollout (Figure 8e).

The version of GraphCast from NVIDIA's Earth2MIP gives reasonable results for just the first 1.5 months (Figure 8c), while that from DeepMind goes bad after a couple weeks (Figure 8g). The SFNO Earth2MIP model (FourCastNetv2-small) shows essentially no drift over a full year (Figure 8d), but it does not follow the annual cycle because it neglects changes in solar forcing. Some artifacts (horizontal stripes) are visible near the south pole within a month and at the north pole much later in the simulation. In contrast, the SFNO Makani model (Figure 8h) includes solar zenith angle as an input field, and it does follow the annual cycle reasonably well. On balance, the performance of the SFNO Makani model is roughly similar to our DLWP-HPX model; it has larger errors near the poles, but less drift in the tropics.

In an ablation study (not shown), we investigated the effect of the top-of atmosphere solar forcing input on the 365-day DLWP-HPX rollout by training a model that did not receive solar forcing input. In that case, the model still generated a stable forecast over the entire rollout period, but did not produce the full annual cycle. Interestingly, that simulation did roughly approximate the transition from summer into a perpetual autumn.

One qualitative way to appreciate the ability of our model to retain realistic weather patterns in a 1,442-step rollout is illustrated by comparing a 360.5 days simulation initialized on 1 April 2017 (with 3-hr resolution) and the corresponding 27 March 2018 reanalysis in Figure 9. The roughly 1-year lead time is well beyond the limits of atmospheric predictability, so there is no reason to expect a close match between simulation and reanalysis. The 360.5-day simulation time was chosen to display the simulated strong low-pressure center in the northeastern Pacific. The intensity of the system is typical for strong systems in our simulation, but its lowest Z_{1000} heights are about 40 m higher than those in the strongest systems periodically appearing in the ERA5 reanalysis. Lower-amplitude signals also appear in the Z_{1000} field, which is somewhat less than 50 m too low in the tropics. On balance, the overall character of this late-March weather pattern is quite plausible. In some models that use latitude-longitude meshes, obvious errors at the poles can show up in as little as 10 autoregressive steps (Bonev et al., 2023, Figure 4). As evident in Figure 9, no artifacts are apparent in the vicinity of the North Pole after 1,442 autoregressive steps.

A more quantitative assessment of any tendency of our model to distort the atmospheric state by damping or amplifying mid-latitude perturbations at different wavelengths is provided by the plots of the Z_{500} power spectral density around 45°N in Figure 10. These spectra are averaged over 208 biweekly forecasts from the 2017 to 2018

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

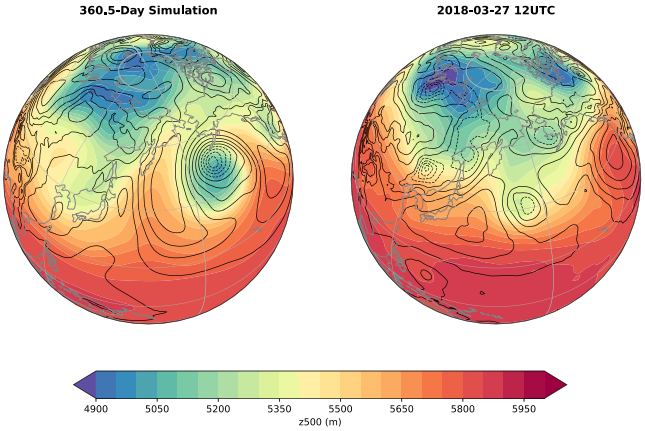


Figure 9. Z_{500} (color fill: 50 dam contour interval) and Z_{1000} (black contours: 40 m interval) for a free-running 360.5-day simulation (1,442 autoregressive steps) and the corresponding ERA5 reanalysis for 00 UTC on 27 March 2018. Dashed black lines indicate values of $Z_{1000} \leq 40$ m (corresponding to sea-level pressures less than roughly 1,008 hPa).

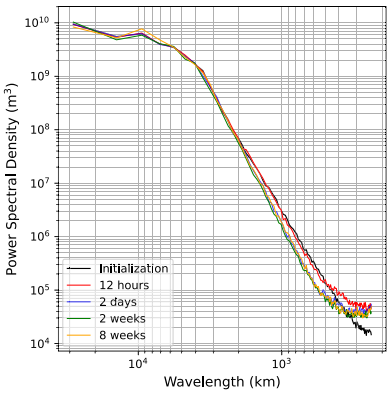


Figure 10. One dimensional power spectral density of the Z_{500} field around the 45°N latitude, averaged over 208 bi-weekly forecasts from 2017 to 2018 at: initialization (black), and at forecast lead times of 12 hr, 2 days, 2, and 8 weeks.

test set for which the RMSE and ACC were plotted in Figure 4. The initial spectrum in black represents the average state of the atmosphere in the ERA5 reanalysis.

Twelve hr (2 recursive steps) after initialization there is very little change in the spectra for wavelengths λ longer than 500 km (roughly 5 grid intervals), but the power in the shorter waves is amplified. Over the next 36 hr, there is a gradual reduction in the amplitude at wavelengths $\lambda < 1{,}800$ km to yield a spectrum that is somewhat damped over the interval $380 < \lambda < 1{,}800$ km and amplified at the shortest wavelengths. Surprisingly, the spectral distribution at 2 days remains essentially unchanged throughout the subsequent autoregressive rollout at least out to sub-seasonal-forecast lead times of 8 weeks (244 steps), which is consistent with the impression obtained by examining images such as those in Figure 9.

What does the deviation of the spectral power from the correct ERA5 curve imply about the ability of the model to approximate a true atmospheric state? As part of the answer, important quantitative points of reference are the RMSE and ACC errors for Z_{500} at day 2 plotted in Figure 4. The day-2 global RMSE error over the same set of forecasts and verifications for which spectra are plotted in Figure 10 is about 17 m; the ACC is negligibly different from the correct value of 1.0. These values represent upper bounds on the 2-day forecast error that might be produced exclusively by the spectral distortion of the Z_{500} field because other factors also contribute to the RMSE and ACC error, such as incorrectly approximating the speed and direction at which features propagate. Of course there is no deterministic predictability at 8-week forecast lead times, but since the 8-week spectrum in Figure 10 is essentially identical to that at

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

2 days, the DLWP-HPX 8-week forecasts need not be farther from some realizable atmospheric state than what is suggested by the modest 2-day Z_{500} errors in Figures 4a and 4d.

5. Conclusion

We have presented an improved CNN-based DLWP-HPX model that stably forecasts atmospheric evolution over a full 1-year cycle using a very limited set of prognostic variables. The number of actual degrees of freedom characterizing predictable atmospheric states at forecast lead times beyond 3–5 days is not known, but is far less than the total number of prognostic variables carried at every grid cell in state-of-the-art NWP models. Here, we have demonstrated that realistic atmospheric simulations can be performed using just seven prognostic variables above each cell on a HEALPix mesh with 110 km between the nodes.

The HEALPix mesh (Gorski et al., 2005) has been used in astronomy for almost two decades, but has previously seen very little use in atmospheric science. The mesh covers the sphere with a hierarchical grid of equal-area cells uniformly spaced along circles at constant latitudes. A particularly important advantage of the HEALPix mesh for weather forecasting with CNNs is that it is an "east to the right" mesh, that is, east has the same orientation in every HEALPix cell. Weather systems tend to travel west-to-east in mid- and high-latitudes and both east-to-west (tropical cyclones) or west-to-east (Madden-Julian Oscillation, convectively coupled Kelvin waves) in the tropics. The kernel weights in our convolutional stencils can more economically learn this behavior than on our previous cubed sphere mesh in which the eastward orientation across the stencil varies with longitude, particularly on the polar faces. More importantly, because all cells have the same east-to-the-right orientation, we do not need to train separate sets of convolution filters for the equatorial and polar regions. Thus, a HEALPix model with the same total number of trainable parameters as a cubed sphere can employ twice as many filter weights as that used for cubed sphere. Although switching from a cubed sphere mesh with 64×64 cells on each of the six faces to a HEALPix mesh with 32×32 cells on each of the 12 faces reduces the total number of grid points covering the sphere by half, it increases the time over which the Z_{500} RMSE remains below 40 m by almost 1/2 days at a 4-day forecast lead time (Figure 5).

Two other significant improvements to our model architecture were obtained by adding recursion via GRUs and by inverting the standard way channel depth is refined at deeper layers in the U-Net. In contrast to the original U-Net architecture Ronneberger et al. (2015), our channel depth halves instead of doubles as the spatial resolution is also halved in each successively deeper U-Net layer. This allows the model to devote more trainable parameters to describing the wide variety of fine-scale weather patterns while using comparatively fewer parameters to describe the simpler set of global weather patterns. Although this modification pushes the U-Net toward the basic ResNet architecture (He et al., 2016), we find the deeper U-Net layers continue to provide significant skill to the forecasts.

Additional modest improvements were implemented by switching to the GELU activation function and to 2×2 transposed strided convolutions when up-sampling; by increasing the total number of trainable parameters from 2.7 to 9.8 M, adding the Z_{250} field, increasing the resolution to HPX64, and increasing the time resolution to 3 hr (which gives us a 6 hr time step). The benefits of 3-hr time resolution were only realized when the model included the GRUs. The 3-hr time resolution gives a good forecast of the daily cycle of surface temperature, and the model also learns the difference in the range of that cycle between regions of tropical forest and desert without geo-specific input data.

Finally, we replaced the pairs of successive convolutions in Weyn et al. (2020) with modified ConvNeXt blocks. The switch to the ConvNeXt blocks was only advantageous at higher resolutions, where in addition to improving accuracy, it reduced the memory footprint.

At 1-week forecast lead time, the resulting model is roughly 1 day behind the ECMWF IFS S2S forecast error in Z_{500} RMSE and 1.5 days behind in ACC. Our statistics are worse than those for Pangu-Weather (Bi et al., 2023) and GraphCast (Lam et al., 2022), both of which provide Z_{500} RMSE and ACC forecasts at $0.25° \times 0.25°$ resolution that are superior to the deterministic ECMWF IFS high-resolution model averaged to the same $0.25° \times 0.25°$ grid. Despite having less accuracy in medium range forecasts, our model can be recursively stepped forward to generate better 500 hPa forecasts over seasonal and 1-year rollouts than GraphCast and Pangu-Weather. It is also superior to the SFNO version of FourCastNetv2 currently on NVIDIA Earth2MIP, though

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

it behaves similarly to the recently checkpointed version of SFNO Makani. Realistic low pressure systems and upper-level trough and ridge patterns continue to be generated by our model at the end of the 1-year rollout.

Deep learning models for weather forecasting are evolving rapidly, with important advancements using a wide variety of architectures. A common methodology in atmospheric science research involves the investigation of some phenomena using a hierarchy of models with decreasing complexity, such as GCMs with full physics parameterizations, simpler nonlinear numerical models with minimal parameterizations, and linear models with analytic solutions. Our DLWP-HPX model provides an example of what can be achieved when training a parsimonious model on a server with just 4 NVIDIA A100 GPUs. It may be particularly useful for scientific investigations when it is advantageous to work with a minimal set of unknown variables to more concisely characterize sensitivities that might be revealed by techniques such as backpropagation with respect to loss functions customized for analysis, as opposed to model training (Ebert-Uphoff et al., 2021). As an example, note that the large-scale structure of the atmosphere is represented in our deepest U-Net layer on each time step by 34 latent-state variables on a coarse-resolution (440 km) grid. This information is decoded during each time step, along with finer resolution latent-state data from the skip connections, to give the updated physical state of the global system. We are currently designing classifier modules configured as a follower network to receive this deep latent-state information to explore the low-frequency variability of the atmosphere.

There are many avenues along which our DLPW-HPX model might be improved. One would be to adding additional prognostic fields while carefully examining the resulting performance. Another one would lie in refining the CNN architecture, where the choice of particular inductive biases may be crucial (Thuemmel et al., 2023). A related important aspect of improving the modeled processes might be to incorporate explicit physical constraints, yielding physics-informed differentiable artificial neural networks (Beucler et al., 2021; Shen et al., 2023). Other natural extensions of this work lie in examining the performance of the DLPW-HPX model in ensemble forecasts, which are crucial to sub-seasonal and seasonal prediction and to couple the atmospheric model with the ocean, thus moving toward a deep learning earth system model (Bauer et al., 2023). Preliminary results suggest that coupling our model with a deep learning ocean model that predicts sea surface temperatures (which are not incorporated in the current model) stabilizes the simulations and removes all model drift in multi-decadal rollouts.

Appendix A: Deep Learning on the HEALPix

A1. Seamless Evolution of Location Invariant Kernels

The Hierarchical Equal Area isoLatitude Pixelization (HEALPix) is a partitioning of the sphere that has found wide application in astronomy since it was introduced by Gorski et al. (2005). It divides the sphere into 12 base faces that can be hierarchically subdivided into patches of equal size. A key property for training CNNs on this mesh is the isolatitudinal alignment, that is, patches are aligned along lines of latitude and each patch has the same orientation, which we describe as “east to the right” in Section 4.1.

To contrast and emphasize the difficulty that CNN kernels are facing on the cubed sphere mesh, we plot the lines of constant latitude on the six faces of the cubed sphere and on the 12 faces of the HEALPix in Figure A1. Except for the equator, all lines of constant latitude are bent on the cubed sphere, imposing challenges for a limited set of convolution kernels that must evolve location invariant pattern detectors and functions. For example, weather systems tend to migrate eastward in mid- and high-latitudes, and the kernels need to learn a wider range of behaviors to propagate eastward motions at the top-left versus the bottom-right corners of the polar faces of the cubed sphere face.

On the other hand, lines of constant latitude map to straight lines on the HEALPix mesh. This facilitates the formulation of location-invariant convolutional kernels for the propagation of weather systems, allowing the same set of kernels to be used over the entire globe. In contrast to the cubed sphere, it is not necessary to train separate sets of kernels for the equatorial and polar faces. Therefore, without increasing the model's total number of trainable parameters, the convolutional kernels on the HEALPix mesh can accommodate more latent layers than on the cubed sphere.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

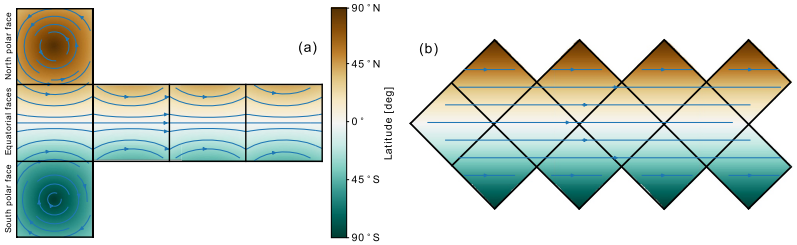


Figure A1. Lines of latitudes depicted as blue streamline arrows on the cubed sphere (a) and on the HEALPix (b). While the lines corresponding to constant eastward motion describe arcs of different radii on the cubed sphere mesh, the same motion translates to straight lines on the HEALPix mesh.

A2. Technical Implementation Details

Since deep learning libraries are optimized for image processing tasks, we consider each of the HEALPix's 12 base faces as a regular two-dimensional tensor, that is, we interpret the sphere as a composition of 12 images (cf., Figures 1 and A2).

To simulate the spatial propagation of dynamics beyond individual faces, such that weather patterns can evolve globally on the sphere, we implement custom padding operations to concatenate the relevant information of all neighboring faces to each respective face of interest.

Figure A2 showcases our planet's coastlines projected on the HEALPix faces in (a) and outlines the spatial organization of the 12 faces in (b). The arrangement of neighboring faces is exemplarily detailed for the northern (N) and southern (S) hemisphere, as well as for the equatorial faces (E). To simulate the neighborhood of, say, face E3, the face N2 must be concatenated to the left of E3, while face S3 is concatenated to the right. On the northern and southern hemispheres, neighboring faces are partially required to be rotated, as indicated in Figures A2c–A2e.

A particular case occurs in the north and south corners of the tropical faces, where no natural neighbor exists—cf., Figures 1 and A2f for an illustration. To simulate the ninth neighbor of the respective corner, we interpolate the values from the according faces on the northern/southern hemisphere, by simply averaging the two corresponding values and writing the result in the simulated neighboring face. For example, to simulate the top left neighboring face of E3, we average the respective values from N2 and N3, as detailed by the straight red arrows in Figure A2g. Values that do not lie on the main diagonal of the simulated face are not required to be interpolated, but are copied from the adjacent faces instead, denoted by the curved red arrows in Figure A2g. The exemplary corner padding shows the case for the application of a 3×3 kernel with dilation of 1 or 2. Note that a 5×5 kernel could be applied in the same way. Importantly, the padding should not extend one neighboring face, which depends on the resolution of the HEALPix mesh and the configuration of the applied convolution (kernel size and dilation). Otherwise, a hierarchy of padding operations would be required to be implemented and considered.

AGU ADVANCING EARTH AND SPACE SCIENCES **Journal of Advances in Modeling Earth Systems** 10.1029/2023MS004021

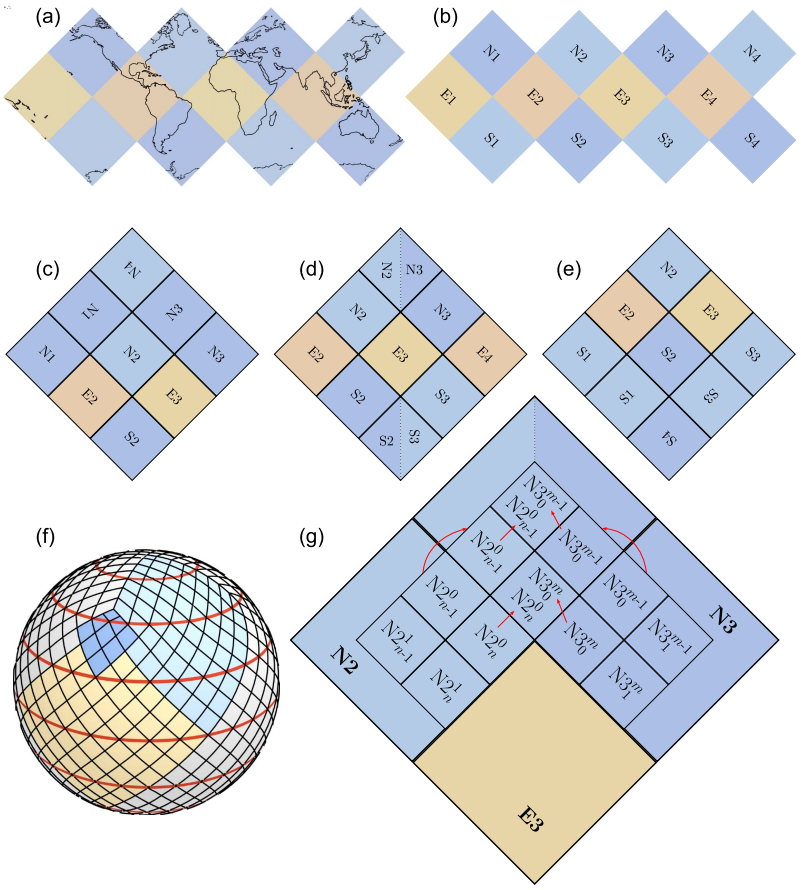


Figure A2. 2D HEALPix face arrangement and padding. (a) Depicts the distribution of coastlines over the 12 HEALPix faces. (b) Enumerates the 12 faces of the HEALPix with each four faces on the northern and southern hemisphere and around the equator. (c)–(e): Exemplary alignment and rotations of neighboring faces before applying the padding operation on northern (c), equatorial (d), and southern faces (e). (f) Emphasizes the special corner case, which is detailed in (g) to visualize the padding. The missing corner pixel is filled by averaging the two values from the adjacent cells (row and column indices of each cell displayed as super- and subscripts, respectively).

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Data Availability Statement

Instructions for data preparation, training, and a trained model for inference, are available online (Karlbauer et al., 2024). In addition, PyTorch code for training the DLWP-HPX model is also available in NVIDIA's modulus repository (NVIDIA, 2024c). All spherical shells of data from ERA5 (Hersbach et al., 2020) were downloaded from Copernicus, where variables on various constant pressure levels (Hersbach et al., 2018a), such as Z_{500} or T_{850}, and variables on single levels (Hersbach et al., 2018b), such as T_{2m} or $TCWV$, are available to the public. To generate 1-year rollouts for Pangu-Weather, GraphCast, and FourCastNet2 (SFNO), as plotted in Figure 8, we considered the respective public repositories with the pretrained model weights. More concretely, we generated the SFNO Earth2MIP (fcnv2_sm) and GraphCast Earth2MIP (graphcast) forecasts with NVIDIA's earth2mip package, specifically developing a custom script for long rollouts (NVIDIA, 2024b). Checkpoints for the SFNO Makani forecast may be found in the NVIDIA NGC catalog (NVIDIA, 2024a). Interestingly, the original GraphCast DeepMind code base (Google, 2024c) produced slightly different results and saturated even faster than the Earth2MIP version, which might result from different random seeds. For the DeepMind version of GraphCast, we downloaded the model weights (Google, 2024a) provided through their repository. Pangu-Weather forecasts in 24 and 3 hr resolution—with respective checkpoint files for the 24 and 3 hr models—were generated by using the original repository (Pangu-Weather, 2024).

Acknowledgments

We would like to thank Mauro Bisson from NVIDIA Corp. for providing optimized CUDA kernels for the HEALPix padding implementation, and Jonathan Weyn who previously implemented a code base on which this work was built. We thank Peter Düben, Imme Ebert-Uphoff, and a third anonymous reviewer for encouraging us to generate and compare the 1-year rollouts for other state-of-the-art DLWP methods and for other valuable suggestions. This work received funding from Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2064—390727645 and from the Office of Naval Research under Grants N00014-21-1-2827 and N00014-22-1-2807. We thank the Deutscher Akademischer Austauschdienst (DAAD, German Academic Exchange Service) as well as the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer. Nathaniel was supported by a National Defense Science and Engineering Graduate Fellowship. We are grateful to NVIDIA and Stan Posey for the donation of A100 GPU cards. This research was additionally supported by a grant from the NVIDIA Applied Research Accelerator Program and utilized an NVIDIA DGX-100 Workstation. Moreover, this work benefited substantially from the barrier-free high quality ERA5 data set provided by the ECMWF. Open Access funding enabled and organized by Projekt DEAL.

References

Ballas, N., Yao, L., Pal, C., & Courville, A. (2015). Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bauer, P., Dueben, P., Chantry, M., Doblas-Reyes, F., Hoefler, T., McGovern, A., & Stevens, B. (2023). Deep learning and a changing economy in weather and climate prediction. *Nature Reviews Earth and Environment*, *4*(8), 507–509. https://doi.org/10.1038/s43017-023-00468-z

Bauer, P., Thorpe, A., & Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, *525*(7567), 47–55. https://doi.org/10.1038/nature14956

Benjamin, S. G., Brown, J. M., Brunet, G., Lynch, P., Saito, K., & Schlatter, T. W. (2019). 100 years of progress in forecasting and NWP applications. *Meteorological Monographs*, *59*, 13.1–13.67. https://doi.org/10.1175/amsmonographs-d-18-0020.1

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., & Gentine, P. (2021). Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, *126*(9), 098302. https://doi.org/10.1103/physrevlett.126.098302

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., & Tian, Q. (2023). Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, *619*(7970), 533–538. https://doi.org/10.1038/s41586-023-06185-3

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., & Anandkumar, A. (2023). Spherical fourier neural operators: Learning stable dynamics on the sphere. *arXiv preprint arXiv:2306.03838*.

Charney, J. G., Fjörtoft, R., & Neumann, J. V. (1950). Numerical integration of the barotropic vorticity equation. *Tellus*, *2*(4), 237–254. https://doi.org/10.1111/j.2153-3490.1950.tb00336.x

Chen, K., Han, T., Gong, J., Bai, L., Ling, F., Luo, J.-J., et al. (2023). Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2020). An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dueben, P. D., & Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, *11*(10), 3999–4009. https://doi.org/10.5194/gmd-11-3999-2018

Ebert-Uphoff, I., Lagerquist, R., Hilburn, K., Lee, Y., Haynes, K., Stock, J., et al. (2021). CIRA guide to custom loss functions for neural networks in environmental sciences—Version 1. *arXiv preprint arXiv:2106.09757*.

ECMWF. (2024a). ECMWF model description. Retrieved from https://confluence.ecmwf.int/display/S2S/ECMWF+model+description

ECMWF. (2024b). S2S, ECMWF, Realtime, Daily averaged [Dataset]. Retrieved from https://apps.ecmwf.int/datasets/data/s2s-realtime-daily-averaged-ecmf/levtype=sfc/type=cf/

Google. (2024a). Checkpoint for graphcast model. Retrieved from https://storage.googleapis.com/dm_graphcast/params/GraphCast/%20-/%20ERA5/%201979-2017/%20-/%20resolution/%200.25/%20-/%20pressure/%20levels/%2037/%20-/%20mesh/%202to6/%20-/%20precipitation/%20input/%20and/%20output.npz

Google. (2024b). Deterministic scores for weatherbench2 models. Retrieved from https://sites.research.google/weatherbench/deterministic-scores/

Google. (2024c). Graphcast model implementation in jax (Software version 0.1). Retrieved from https://github.com/google-deepmind/graphcast

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, 2005* (Vol. 2, pp. 729–734).

Gorski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., & Bartelmann, M. (2005). Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, *622*(2), 759–771. https://doi.org/10.1086/427976

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., & Catanzaro, B. (2021). Efficient token mixing for transformers via adaptive fourier neural operators. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., et al. (2018a). ERA5 hourly data on pressure levels from 1940 to present [Dataset]. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*. https://doi.org/10.24381/cds.bd0915c6

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., et al. (2018b). ERA5 hourly data on pressure levels from 1940 to present [Dataset]. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*. https://doi.org/10.24381/cds.adbb2d47

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., et al. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999–2049. https://doi.org/10.1002/qj.3803

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, Y., Chen, L., Wang, Z., & Li, H. (2022). Swinvrnn: A data-driven ensemble forecasting model via learned distribution perturbation. *arXiv preprint arXiv:2205.13158*.

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., et al. (2020). Unet 3+: A full-scale connected UNET for medical image segmentation. In *ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1055–1059).

Karlbauer, M., Cresswell-Clay, N., & Kurth, T. (2024). Deep learning weather prediction on HEALPix. Zenodo (Software version 0.2.0). https://doi.org/10.5281/zenodo.12200496

Keisler, R. (2022). Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Krachmalnicoff, N., & Tomasi, M. (2019). Convolutional neural networks on the healpix sphere: A pixel-based algorithm and its application to CMB data analysis. *Astronomy and Astrophysics*, *628*, A129. https://doi.org/10.1051/0004-6361/201935211

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., et al. (2022). Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. *arXiv preprint arXiv:2208.05419*.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., et al. (2022). Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012–10022).

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11976–11986).

Lopez-Gomez, I., McGovern, A., Agrawal, S., & Hickey, J. (2022). Global extreme heat forecasting using neural weather models. *arXiv preprint arXiv:2205.10972*.

Lorenz, E. N. (1969). The predictability of a flow which possesses many scales of motion. *Tellus*, *21*(3), 289–307. https://doi.org/10.3402/tellusa.v21i3.10086

Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

NVIDIA. (2024a). Checkpoint for SFNO model. Retrieved from https://catalog.ngc.nvidia.com/orgs/nvidia/teams/modulus/models/sfno_73ch_small/files

NVIDIA. (2024b). Earth-2 model intercomparison project (Software version 0.1.0). Retrieved from https://github.com/NVIDIA/earth2mip/blob/main/examples/utils/workflows/1_year_run.py

NVIDIA. (2024c). Modulus (Software version 0.6.0). Retrieved from https://github.com/NVIDIA/modulus/tree/main/examples/weather/dlwp_healpix

Palmer, T. (2019). The ECMWF ensemble prediction system: Looking back (more than) 25 years and projecting forward 25 years. *Quarterly Journal of the Royal Meteorological Society*, *145*(S1), 12–24. https://doi.org/10.1002/qj.3383

Pangu-Weather. (2024). An official implementation of Pangu-Weather (Software version 1.0). Retrieved from https://github.com/198808xc/Pangu-Weather?tab=readme-ov-file#downloading-trained-models

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., et al. (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*.

Perraudin, N., Defferrard, M., Kacprzak, T., & Sgier, R. (2019). Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astronomy and Computing*, *27*, 130–146. https://doi.org/10.1016/j.ascom.2019.03.004

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*.

Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russel, T., et al. (2023). Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*.

Rasp, S., & Thuerey, N. (2021). Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, *13*(2), e2020MS002405. https://doi.org/10.1029/2020ms002405

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 234–241).

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. https://doi.org/10.1109/tnn.2008.2005605

Scher, S., & Messori, G. (2018). Predicting weather forecast uncertainty with machine learning. *Quarterly Journal of the Royal Meteorological Society*, *144*(717), 2830–2841. https://doi.org/10.1002/qj.3410

Scher, S., & Messori, G. (2019). Weather and climate forecasting with neural networks: Using GCMs with different complexity as study-ground. *Geoscientific Model Development*, *12*(7), 2797–2809. https://doi.org/10.5194/gmd-12-2797-2019

Shen, C., Appling, A. P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., et al. (2023). Differentiable modelling to unify machine learning and physical models for geosciences. *Nature Reviews Earth and Environment*, *4*(8), 552–567. https://doi.org/10.1038/s43017-023-00450-9

Thuemmel, J., Karlbauer, M., Otte, S., Zarfl, C., Martius, G., Ludwig, N., et al. (2023). Inductive biases in deep learning models for weather prediction. *arXiv preprint arXiv:2304.04664*.

Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic Geography*, *46*(sup1), 234–240. https://doi.org/10.2307/143141

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Journal of Advances in Modeling Earth Systems 10.1029/2023MS004021

Vitart, F. (2004). Monthly forecasting at ECMWF. *Monthly Weather Review*, *132*(12), 2761–2779. https://doi.org/10.1175/MWR2826.1

Weigel, A. P., Baggenstos, D., Liniger, M. A., Vitart, F., & Appenzeller, C. (2008). Probabilistic verification of monthly temperature forecasts. *Monthly Weather Review*, *136*(12), 5162–5182. https://doi.org/10.1175/2008MWR2551.1

Weyn, J. A., Durran, D. R., & Caruana, R. (2019). Can machines learn to predict weather? Using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2680–2693. https://doi.org/10.1029/2019ms001705

Weyn, J. A., Durran, D. R., & Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, *12*(9), e2020MS002109. https://doi.org/10.1029/2020ms002109

Weyn, J. A., Durran, D. R., Caruana, R., & Cresswell-Clay, N. (2021). Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, *13*(7), e2021MS002502. https://doi.org/10.1029/2021ms002502

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support* (pp. 3–11). Springer.

Bibliography

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, **6**, 14410–14430.

Amari, S.-I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, **100**(11), 1197–1206.

Ballas, N., Yao, L., Pal, C., and Courville, A. (2015). Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*.

Barnett, A. G., Van Der Pols, J. C., and Dobson, A. J. (2005). Regression to the mean: what it is and how to deal with it. *International journal of epidemiology*, **34**(1), 215–220.

Basdevant, C., Deville, M., Haldenwang, P., Lacroix, J., Ouazzani, J., Peyret, R., Orlandi, P., and Patera, A. (1986). Spectral and finite difference solutions of the burgers equation. *Computers & Fluids*, **14**(1), 23–41.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., *et al.* (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bauer, H.-S., Muppa, S. K., Wulfmeyer, V., Behrendt, A.,

Warrach-Sagi, K., and Späth, F. (2020). Multi-nested wrf simulations for studying planetary boundary layer processes on the turbulence-permitting scale in a realistic mesoscale environment. *Tellus A: Dynamic Meteorology and Oceanography*, **72**(1), 1–28.

Bauer, P., Thorpe, A., and Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, **525**(7567), 47–55.

Bauer, P., Dueben, P., Chantry, M., Doblas-Reyes, F., Hoefler, T., McGovern, A., and Stevens, B. (2023). Deep learning and a changing economy in weather and climate prediction. *Nature Reviews Earth & Environment*, pages 1–3.

Ben-Bouallegue, Z., Clare, M. C., Magnusson, L., Gascon, E., Maier-Gerber, M., Janousek, M., Rodwell, M., Pinault, F., Dramsch, J. S., Lang, S. T., *et al.* (2023). The rise of data-driven weather forecasting. *arXiv preprint arXiv:2307.10128*.

Berahas, A. S., Nocedal, J., and Takác, M. (2016). A multi-batch l-bfgs method for machine learning. *Advances in Neural Information Processing Systems*, **29**.

Beucler, T., Rasp, S., Pritchard, M., and Gentine, P. (2019). Achieving conservation of energy in neural network emulators for climate modeling. *arXiv preprint arXiv:1906.06622*.

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., and Gentine, P. (2021). Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, **126**(9), 098302.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. (2023). Accurate medium-range global weather forecasting with 3d neural networks. *Nature*.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill,

E., *et al.* (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Bou-Zeid, E., Anderson, W., Katul, G. G., and Mahrt, L. (2020). The persistent challenge of surface heterogeneity in boundary-layer meteorology: a review. *Boundary-Layer Meteorology*, **177**, 227–245.

Breiman, L. (2001). Random forests. *Machine learning*, **45**, 5–32.

Buizza, R. and Leutbecher, M. (2015). The forecast skill horizon. *Quarterly Journal of the Royal Meteorological Society*, **141**(693), 3366–3382.

Butz, M. V., Bilkey, D., Humaidan, D., Knott, A., and Otte, S. (2019). Learning, planning, and control in a monolithic neural event inference architecture. *Neural Networks*, **117**, 135–144.

Chelton, D. B. and Xie, S.-P. (2010). Coupled ocean-atmosphere interaction at oceanic mesoscales. *Oceanography*, **23**(4), 52–69.

Chen, K., Han, T., Gong, J., Bai, L., Ling, F., Luo, J.-J., Chen, X., Ma, L., Zhang, T., Su, R., *et al.* (2023). Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*.

Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

Chen, X., Hsieh, C.-J., and Gong, B. (2021). When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ciurletti, M., Traub, M., Karlbauer, M., Butz, M. V., and Otte, S. (2021). Signal denoising with recurrent spiking neural networks and active tuning. In *International Conference on Artificial Neural Networks*, pages 220–232. Springer.

Cowan, I., Farquhar, G., and Jennings, D. (1977). Integration of activity in the higher plant. *Stomatal function in relation to leaf metabolism and environment. Cambridge Univ Press, Cambridge*, pages 471–505.

De Bézenac, E., Pajot, A., and Gallinari, P. (2019). Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, **2019**(12), 124009.

DeepMind, G. (2019). Alphazero: Shedding new light on chess, shogi, and go.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., *et al.* (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dueben, P. D. and Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, **11**(10), 3999–4009.

Durran, D. R. and Frierson, D. M. W. (2013). Condensation, atmospheric motion, and cold beer. *Physics Today*, **66**(4), 74–75.

Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., Gazen, C., Hickey, J., Bell, A., and Kalchbrenner, N. (2021). Skillful Twelve Hour Precipitation Forecasts using Large Context Neural Networks. arXiv:2111.07470 [physics].

Fayek, H. M., Cavedon, L., and Wu, H. R. (2020). Progressive

learning: A deep learning framework for continual learning. *Neural Networks*, **128**, 345–357.

Fei, H. and Tan, F. (2018). Bidirectional grid long short-term memory (bigridlstm): A method to address context-sensitivity and vanishing gradient. *Algorithms*, **11**(11), 172.

Gao, Z., Shi, X., Han, B., Wang, H., Jin, X., Maddix, D., Zhu, Y., Li, M., and Wang, Y. (2023). Prediff: Precipitation nowcasting with latent diffusion models. *arXiv preprint arXiv:2307.10422*.

Gentine, P., Beucler, T., Pritchard, M. S., and Eyring, V. (2020). Incorporating physical knowledge in machine learning parameterizations of convection. In *AGU Fall Meeting Abstracts*, volume 2020, pages A056–02.

Guen, V. and Thome, N. (2020). Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11474–11484.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

Hansen, D., Maddix, D. C., Alizadeh, S., Gupta, G., and Mahoney, M. W. (2023). Learning physical models that can respect conservation laws. *arXiv preprint arXiv:2302.11002*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers,

D., *et al.* (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, **146**(730), 1999–2049.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, **91**(1), 31.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., *et al.* (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hoffmann, S. and Lessig, C. (2023). Atmodist: Self-supervised representation learning for atmospheric dynamics. *Environmental Data Science*, **2**, e6.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, **2**(5), 359–366.

Horuz, C. C., Karlbauer, M., Praditia, T., Butz, M. V., Oladyshkin, S., Nowak, W., and Otte, S. (2022). Infering boundary conditions in finite volume neural networks. In *International Conference on Artificial Neural Networks*, pages 538–549. Springer.

Horuz, C. C., Karlbauer, M., Praditia, T., Butz, M. V., Oladyshkin, S., Nowak, W., and Otte, S. (2023). Physical domain reconstruction with finite volume neural networks. *Applied Artificial Intelligence*, **37**(1), 2204261.

Kaack, L. H., Donti, P. L., Strubell, E., Kamiya, G., Creutzig, F., and Rolnick, D. (2022). Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, pages 1–10.

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., and Butz, M. V. (2020a). A distributed neural network architecture for robust non-linear spatio-temporal prediction. *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Karlbauer, M., Otte, S., Lensch, H. P., Scholten, T., Wulfmeyer, V., and Butz, M. V. (2020b). Inferring, predicting, and denoising causal wave dynamics. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I 29*, pages 566–577. Springer.

Karlbauer, M., Menge, T., Otte, S., Lensch, H. P., Scholten, T., Wulfmeyer, V., and Butz, M. V. (2021). Latent state inference in a spatiotemporal generative model. In *International Conference on Artificial Neural Networks*, pages 384–395. Springer.

Karlbauer, M., Praditia, T., Otte, S., Oladyshkin, S., Nowak, W., and Butz, M. V. (2022). Composing partial differential equations with physics-aware neural networks. In *International Conference on Machine Learning*, pages 10773–10801. PMLR.

Karlbauer, M., Cresswell-Clay, N., Durran, D. R., Moreno, R. A., Kurth, T., Bonev, B., Brenowitz, N., and Butz, M. V. (2024).

Advancing parsimonious deep learning weather prediction using the healpix mesh. *Journal of Advances in Modeling Earth Systems*, **16**(8), e2023MS004021.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, **3**(6), 422–440.

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., *et al.* (2021). Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, **379**(2194), 20200093.

Kautz, H. (2022). The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Magazine*, **43**(1), 105–125.

Kautz, L.-A., Martius, O., Pfahl, S., Pinto, J. G., Ramos, A. M., Sousa, P. M., and Woollings, T. (2022). Atmospheric blocking and weather extremes over the euro-atlantic sector–a review. *Weather and climate dynamics*, **3**(1), 305–336.

Kendon, E. J., Stratton, R. A., Tucker, S., Marsham, J. H., Berthou, S., Rowell, D. P., and Senior, C. A. (2019). Enhanced future changes in wet and dry extremes over africa at convection-permitting scale. *Nature communications*, **10**(1), 1794.

Klaasen, G. and Troy, W. (1984). Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, **44**(1), 96–110.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, **118**(21).

Korhonen, J. and You, J. (2012). Peak signal-to-noise ratio revisited: Is simple beautiful? In *2012 Fourth International Workshop on Quality of Multimedia Experience*, pages 37–38. IEEE.

Kraus, E. B. and Businger, J. A. (1994). *Atmosphere-ocean interaction*, volume 27. Oxford University Press.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, **25**.

Lagerquist, R. and Ebert-Uphoff, I. (2022). Can we integrate spatial verification methods into neural-network loss functions for atmospheric science? *arXiv preprint arXiv:2203.11141*.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., *et al.* (2022). Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*.

Lea, C., Vidal, R., Reiter, A., and Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 47–54. Springer.

Lenton, T. M., Rockström, J., Gaffney, O., Rahmstorf, S., Richardson, K., Steffen, W., and Schellnhuber, H. J. (2019). Climate tipping points—too risky to bet against. *Nature*, **575**(7784), 592–595.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Lienen, M. and Günnemann, S. (2022). Learning the dynamics of physical systems from sparse observations with finite element networks. *arXiv preprint arXiv:2203.08852*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Ph.D. thesis, Master's Thesis (in Finnish), Univ. Helsinki.

Liu, Z. and Alexander, M. (2007). Atmospheric bridge, oceanic tunnel, and global climatic teleconnections. *Reviews of Geophysics*, **45**(2).

Lorenz, E. N. (1963a). Deterministic nonperiodic flow. *Journal of atmospheric sciences*, **20**(2), 130–141.

Lorenz, E. N. (1963b). The predictability of hydrodynamic flow. *Trans. NY Acad. Sci*, **25**(4), 409–432.

Luc, P., Clark, A., Dieleman, S., Casas, D. d. L., Doron, Y., Cassirer, A., and Simonyan, K. (2020). Transformation-based adversarial video prediction on large-scale data. *arXiv preprint arXiv:2003.04035*.

Lupo, A. R. (2021). Atmospheric blocking events: A review. *Annals of the New York Academy of Sciences*, **1504**(1), 5–24.

Lynch, P. (2006). *The emergence of numerical weather prediction: Richardson's dream*. Cambridge University Press.

Malik, S., Pal, S. C., Sattar, A., Singh, S. K., Das, B., Chakrabortty, R., and Mohammad, P. (2020). Trend of extreme rainfall events using suitable global circulation model to combat

the water logging condition in kolkata metropolitan area. *Urban Climate*, **32**, 100599.

Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, **5**, 115–133.

Meehl, G. A., Boer, G. J., Covey, C., Latif, M., and Stouffer, R. J. (2000). The coupled model intercomparison project (cmip). *Bulletin of the American Meteorological Society*, **81**(2), 313–318.

Moukalled, F., Mangani, L., and Darwish, M. (2016). *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition.

Nakamura, N. and Huang, C. S. (2018). Atmospheric blocking as a traffic jam in the jet stream. *Science*, **361**(6397), 42–47.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., and Grover, A. (2023). Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*.

Nowak, W. (2000). *Age determination of a TCE source zone using solute transport profiles in an underlying clayey aquitard*. University of Waterloo.

Nowak, W. and Guthke, A. (2016). Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, **18**(11).

Otte, S., Schmitt, T., Friston, K., and Butz, M. V. (2017). Inferring adaptive goal-directed behavior within recurrent neural networks. In *Artificial Neural Networks and Machine Learning–ICANN 2017: 26th International Conference on Artificial Neu-*

ral Networks, Alghero, Italy, September 11-14, 2017, Proceedings, Part I 26, pages 227–235. Springer.

Otte, S., Karlbauer, M., and Butz, M. V. (2020). Active tuning. *arXiv preprint arXiv:2010.03958*.

Palmer, T. (2019). The ecmwf ensemble prediction system: Looking back (more than) 25 years and projecting forward 25 years. *Quarterly Journal of the Royal Meteorological Society*, **145**, 12–24.

Palmer, T. and Hagedorn, R. (2006). *Predictability of weather and climate*. Cambridge University Press.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., *et al.* (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*.

Phillips, N. A. and Shukla, J. (1973). On the strategy of combining coarse and fine grid meshes in numerical weather prediction. *Journal of Applied Meteorology and Climatology*, **12**(5), 763–770.

Post, A. K. and Knapp, A. K. (2020). The importance of extreme rainfall events and their timing in a semi-arid grassland. *Journal of Ecology*, **108**(6), 2431–2443.

Praditia, T., Karlbauer, M., Otte, S., Oladyshkin, S., Butz, M. V., and Nowak, W. (2022). Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network. *Water Resources Research*, **58**(12), e2022WR033149.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differ-

ential equations. *Journal of Computational Physics*, **378**, 686–707.

Rangapuram, S. S., Kapoor, S., Nirwan, R. S., Mercado, P., Januschowski, T., Wang, Y., and Bohlke-Schneider, M. (2023). Coherent probabilistic forecasting of temporal hierarchies. In *International Conference on Artificial Intelligence and Statistics*, pages 9362–9376. PMLR.

Rasp, S. and Lerch, S. (2018). Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, **146**(11), 3885–3900.

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thuerey, N. (2020). Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, **12**(11), e2020MS002203.

Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., *et al.* (2021). Skilful precipitation nowcasting using deep generative models of radar. *Nature*, **597**(7878), 672–677.

Reda, F. A., Liu, G., Shih, K. J., Kirby, R., Barker, J., Tarjan, D., Tao, A., and Catanzaro, B. (2018). Sdc-net: Video prediction using spatially-displaced convolution. In *Proceedings of the European conference on computer vision (ECCV)*, pages 718–733.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., *et al.* (2019). Deep learning and process understanding for data-driven earth system science. *Nature*, **566**(7743), 195–204.

Richardson, L. F. (1922). *Weather prediction by numerical process*. Cambridge university press.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Con-

volutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, **65**(6), 386.

Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, **11**(5), 561–580.

Scher, S. and Messori, G. (2018). Predicting weather forecast uncertainty with machine learning. *Quarterly Journal of the Royal Meteorological Society*, **144**(717), 2830–2841.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., and Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, **379**(2194), 20200097.

Schymanski, S. J., Roderick, M. L., Sivapalan, M., Hutley, L. B., and Beringer, J. (2008). A canopy-scale test of the optimal water-use hypothesis. *Plant, Cell & Environment*, **31**(1), 97–111.

Shen, C., Appling, A. P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., Baity-Jesi, M., Fenicia, F., Kifer, D., Li, L., *et al.* (2023). Differentiable modelling to unify machine learning and physical models for geosciences. *Nature Reviews Earth & Environment*, pages 1–16.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning

approach for precipitation nowcasting. *Advances in neural information processing systems*, **28**.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, **375**, 1339–1364.

Skok, G. and Roberts, N. (2016). Analysis of fractions skill score properties for random precipitation fields and ecmwf forecasts. *Quarterly Journal of the Royal Meteorological Society*, **142**(700), 2599–2610.

Somerville, R. C. (1987). The predictability of weather and climate. *Climatic Change*, **11**(1-2), 239–246.

Steffen, W., Richardson, K., Rockström, J., Cornell, S. E., Fetzer, I., Bennett, E. M., Biggs, R., Carpenter, S. R., De Vries, W., De Wit, C. A., *et al.* (2015). Planetary boundaries: Guiding human development on a changing planet. *Science*, **347**(6223), 1259855.

Stendel, M., Francis, J., White, R., Williams, P. D., and Woollings, T. (2021). The jet stream and climate change. In *Climate Change*, pages 327–357. Elsevier.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Tesch, T., Kollet, S., and Garcke, J. (2023). Causal deep learning models for studying the earth system. *Geoscientific Model Development*, **16**(8), 2149–2166.

Thuemmel, J., Karlbauer, M., Otte, S., Zarfl, C., Martius, G., Ludwig, N., Scholten, T., Friedrich, U., Wulfmeyer, V., Goswami, B., *et al.* (2023). Inductive biases in deep learning models for weather prediction. *arXiv preprint arXiv:2304.04664*.

Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, **46**(sup1), 234–240.

Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, **237**, 37–72.

Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., and Carver, G. (2017). Single precision in weather forecasting models: An evaluation with the ifs. *Monthly Weather Review*, **145**(2), 495–502.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, **13**(4), 600–612.

Warrach-Sagi, K., Ingwersen, J., Schwitalla, T., Troost, C., Aurbacher, J., Jach, L., Berger, T., Streck, T., and Wulfmeyer, V. (2022). Noah-mp with the generic crop growth model gecros in the wrf model: Effects of dynamic crop growth on land-atmosphere interaction. *Journal of Geophysical Research: Atmospheres*, **127**(14), e2022JD036518.

Weyn, J. A., Durran, D. R., and Caruana, R. (2019). Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, **11**(8), 2680–2693.

Weyn, J. A., Durran, D. R., and Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, **12**(9), e2020MS002109.

Weyn, J. A., Durran, D. R., Caruana, R., and Cresswell-Clay, N. (2021). Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, **13**(7), e2021MS002502.

White, C. J., Carlsen, H., Robertson, A. W., Klein, R. J., Lazo, J. K., Kumar, A., Vitart, F., Coughlan de Perez, E., Ray, A. J., Murray, V., *et al.* (2017). Potential applications of subseasonal-to-seasonal (s2s) predictions. *Meteorological applications*, **24**(3), 315–325.

Wulfmeyer, V., Pineda, J. M. V., Otte, S., Karlbauer, M., Butz, M. V., Lee, T. R., and Rajtschan, V. (2023). Estimation of the surface fluxes for heat and momentum in unstable conditions with machine learning and similarity approaches for the lafe data set. *Boundary-Layer Meteorology*, **186**(2), 337–371.

Yin, Y., Guen, V., Dona, J., Ayed, I., de Bézenac, E., Thome, N., and Gallinari, P. (2020). Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*.

Yin, Y., Le Guen, V., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. (2021). Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, **2021**(12), 124012.

Yuval, J., O'Gorman, P. A., and Hill, C. N. (2021). Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, **48**(6), e2020GL091363.

Zhang, D.-L. (2020). Rapid urbanization and more extreme rainfall events. *Science Bulletin*, **65**(7), 516–518.

Acknowledgements

First of all, this thesis would not have materialized without the initiative of my doctor's father Martin Butz, who literally was a father for me in various aspects. Martin, I enjoy your approachability, your sense for human relations, your positive and supportive mood, I appreciate your interest and curiosity to break the mold, your acuity of thought when contemplating about scientific content, and I thank you for sharing your infectious passion of advancing science for the sake of humanity under consideration of our planetary resources. It has been a pleasure to thrive under your benevolent mentorship.

In the same sense, I express my profound gratitude to Sebastian Otte. Thank you for showing me the way into science and for safely walking me through the jungles of LaTeXwithout drowning in round, square, curly, and otherwise shaped brackets (that's a science of its own), for being an inspiring role model, for animating the entire lab to participate in pull-up sessions, for always spreading a positive mood and sharing your genuine sense of humor, and for your unconditional support both in the professional and private context.

As more and less official members of my thesis advisory committee, I want to thank Hendrik Lensch, Georg Martius, Thomas Scholten, and Volker Wulfmeyer for pleasant and fruitful conversations in status meetings and beyond, and for accompanying me complaisantly through the entire path describing the trajectory of my PhD.

A collaboration of high value rose in the joint work with the Institute for Modelling Hydraulic and Environmental Systems (IWS) at the University of Stuttgart, personified in Wolfgang Nowak, Sergey Oladyshkin, and, in particular, Timothy Praditia. Our joint publications not only extended my horizon into the vast field of physics, but also opened doors for me to various opportunities; and I literally enjoyed our formal meetings with lots of informal and highly speculative content.

Talking about collaborations, I want to thank Dale Durran from the University of Washington, for mentoring me patiently in atmospheric science questions, for giving me opportunities to explore the field of DLWP, for being a researcher as enthusiastic as Martin, and for supporting me not only during my stay in Seattle (by thankfully insisting to lend me a helmet and lights for my daily bicycle rides to the office), but also until the very end of my PhD and beyond. Taking the opportunity, I also want to thank Nathaniel and Erika, as well as Nils and his family for enriching my life within and beyond work in Seattle.

The same thank must be pronounced to Gabriele Messori at Uppsala University, who did not hesitate to host and mentor me at the Department of Earth Sciences, introducing me to a benign and inclusive institute culture that taught me lessons of how to survive winter in Sweden—I would really like to return one day and complete our work on ensembling strategies in DLWP.

During my time with Martin's Neuro-Cognitive Modeling Group in Tübingen, I enjoyed the atmosphere among former and current colleagues, which never appeared competitive but inclusive, appreciative, collaborative, and supportive. I never hesitated coming to the office and having lunch together, discussing new trends in reading groups, and spending time in informal and lively conversations. I am grateful for having been part of such a peaceful and stimulating environment. Particularly, I thank Jannik, Fedor, and

Martin for proofreading and feedback on this thesis.

For financial and mental support, I want to thank the cluster of excellence "Machine Learning: New Perspectives for Science," particularly mentioning Tilman Gocht who had an open ear and advice when I had organizational questions, and the International Max-Planck Research School for Intelligent Systems (IMPRS-IS), enthusiastically organized by Leila Masri, who showed me the value of an inclusive community.

Furthermore, I want to thank Manuela DiPaolo for being a reliable, curious, warm, and supportive secretary, for being like a mother in work-affairs and keeping my back free of bureaucratic hurdles, for giving advice in contract-related questions, and for taking care of my letters during my times abroad.

Finally, I formulate an extraordinary thank you to my family and friends, who never doubted my skill to accomplish my PhD, who never abandoned me, also and particularly during tough times when I had no capacity to give, for standing with me in hard times that I and we went through, and for giving me a sustainable and lasting meaning in life.

Artificial Neural Networks

This thesis explores the potential of machine learning methods for improving weather forecasts. Since weather is considered a spatiotemporal process that evolves over space through time, the thesis first investigates the design choices required for machine learning models to simulate synthetic spatiotemporal processes, such as the two-dimensional wave equation. It then develops a method for analyzing machine learning models that enables the extraction of unknown process-relevant context that parameterizes an observed simulated spatiotemporal process of interest. Relating these extracted factors to physical properties leads the thesis to physics-aware machine learning, where it explores how to fuse process knowledge from physics with the learning ability of artificial neural networks. Given the insights from those investigations, a competitive deep learning weather prediction model is designed to understand which design choices support data-driven algorithms to learn a meaningful function that predicts realistic and stable states of the atmosphere over hundreds of hours, days, and weeks into the future.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

TÜBINGEN
LIBRARY PUBLISHING

9 783989 440241